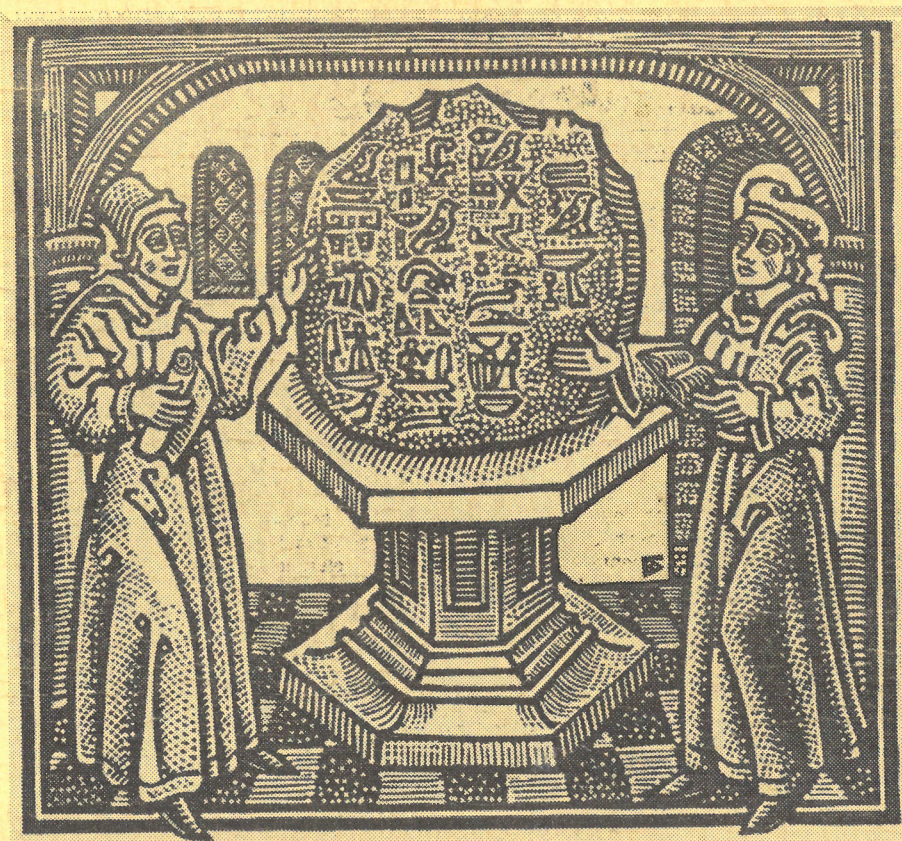


COMMUNICATION SYSTEMS AND COMPUTER NETWORKS

Volume I

Edition 7.1

Pieter Kritzinger

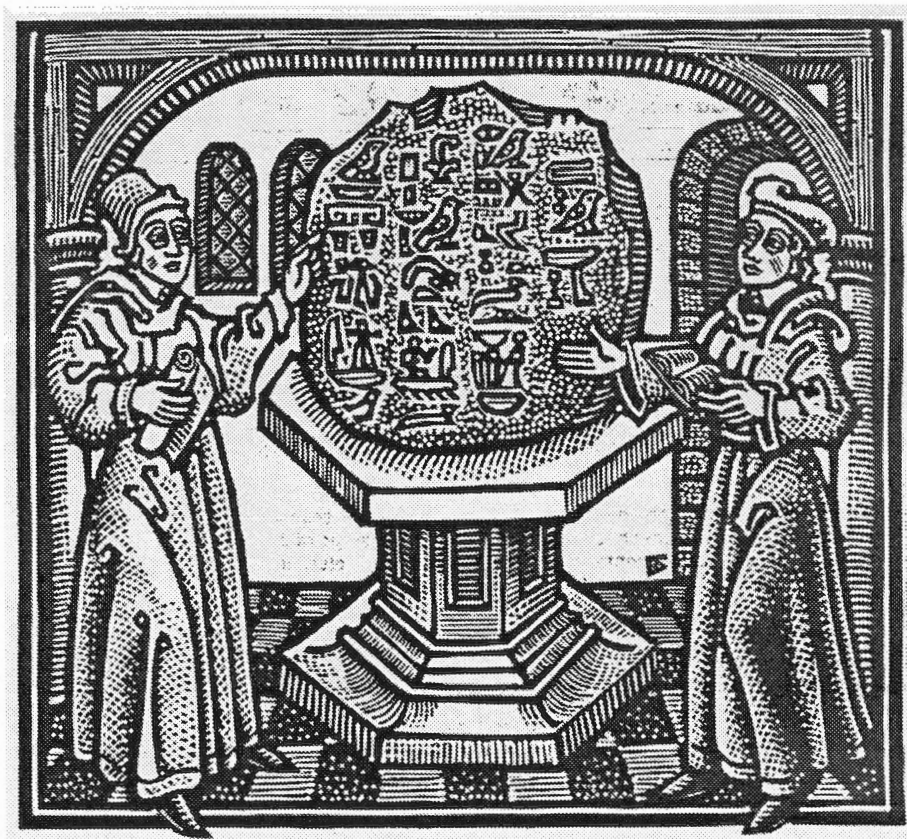


COMMUNICATION SYSTEMS AND COMPUTER NETWORKS

Volume I

Edition 7.1

Pieter Kritzinger



Foreword

Man has always had a desire to exchange information amongst fellow man. Not so long ago, people gathered on the town square to gossip about the affairs of the town and its inhabitants. The Internet has now turned the World into one big electronic town square. As in the early days, the value of the information thus disseminated should be taken with a grain of salt, or at least should be verified before taken seriously. Most individuals seem to forget this age old adage.

This version 7.1 constitutes a major rewrite of the previous version and as the new title indicates, reflects the merging of telephone and computer and other networks into a single telecommunication network which would carry video, data and voice.

No single person can invent the knowledge reflected in these pages and so the content is compiled from many articles, textbooks and my own work. There are many good textbooks on the topic and the one which comes closest to covering the material in these notes is *Understanding Data Communications and Networks* by W.A. Shay and published by the PWS Publishing Company (ISBN 0-534-20244-6). Another good text, *albeit* somewhat academic in nature, is the Third edition of *Computer Networks* by Andrew Tanenbaum (ISBN 0-13-394248-1). The one I would recommend you to buy (as of June 15, 2002) is *Computer Networks and Internets* (Second Edition) by Douglas E Comer, published by Prentice-Hall International, ISBN 0-13-084222-2.

The subject matter of (tele-)communications has not only become vast but much more significant in the world of computing than it was a mere decade ago. Recognition of this is the introduction of the *C* in the previous *IT of Information Technology* to turn it into *Information and Communication Technology (ICT)*. No text can expect to cover all of it unless one wants to leave the reader with a feeling of “knowing nothing about everything”. If the reader discovers that his or her favourite network topic is therefore not even mentioned: that is inevitable.

This first revision of the seventh edition of Volume I of Telecommunication and Computer Networks has three parts. Part I is intended to lay the foundation for a course in the subject and as such could serve as reading material for a few introductory lectures the subject. Part II covers the fundamentals of data communication. It introduces the engineering concepts required to understand both electrical and optical data channels and their properties. Part III concerns the basic protocols required in any network. Starting with the Data Link layer it considers Packet Switching and Media Access Control (MAC) techniques used by the common local networks. The volume ends with a chapter on the Transport Layer. The material covered to this point are required reading for a good undergraduate course in computer networks.

Volume II covers advanced topics such as formal description techniques including SDL, UML, Petrinets and process algebras, network security, network management and advanced network architecture, namely Fibre Distributed Data Interface (FDDI) and Distributed Queue Dual Bus (DQDB). The second volume is intended for a second course in telecommunications and computer networks.

Many generations of students have worked on these notes: The last such student was Oksana Ryndina; to her and all the ones before her, here with my expression of gratitude again. Any constructive critique about the content of these notes will always be most welcome.

Pieter Kritzinger
Dar es Salaam
July 2002

Contents

I	Overview of Telecommunication and Computer Networks	3
1	Basic Concepts	5
1.1	Objectives of this Chapter	5
1.2	Introduction	6
1.3	Evolution of Computer Networks	6
1.4	Electrical Signals	9
1.5	Optical Signals	11
1.6	A Matter of Protocol	11
1.7	Network Architecture	12
1.7.1	Physical layer	14
1.7.2	Data link layer	15
1.7.3	Network layer	15
1.7.4	Transport layer	16
1.7.5	Session layer	16
1.7.6	Presentation layer	17
1.7.7	Application layer	17
1.7.8	Standards and ISO	17
1.8	International Standards Bodies	17
1.9	Exercises	18
II	Transmission Fundamentals	21
2	Electrical Signals	23
2.1	Objectives of this Chapter	23
2.2	Fourier Analysis	25
2.3	Transmission Media	26

2.3.1	Twisted pair	27
2.3.2	Coaxial cable	28
2.4	Channel Characteristics	28
2.4.1	Attenuation	28
2.4.2	Delay distortion	30
2.4.3	Electrical noise	30
2.4.4	Bandwidth limitation	31
2.5	Data encoding	34
2.5.1	Digital Data, Digital Signals	34
2.5.2	Digital Data, Analog Signals	37
2.5.3	Analog Data, Digital Signals	39
2.6	Exercises	40
3	Optic Signals	43
3.1	Objectives of this Chapter	43
3.2	Optic Fibres	44
3.3	Method of Transmission	45
3.4	Optic Media	45
3.4.1	Types of Fibre	46
3.4.2	Optic Signal Generation	48
3.5	Signal Loss in Optic Fibre	49
3.6	Advantages and Disadvantages of Optic Fibres	50
3.7	Exercises	50
4	Wireless Communication	51
4.1	Objectives of this Chapter	51
4.2	Frequency Spectrum	52
4.3	Radio Transmission	53
4.4	Terrestrial Microwave Signals	54
4.5	Satellite Communication	54
4.6	Exercises	56

5	Channels and their Properties	57
5.1	Objectives of this Chapter	57
5.2	Introduction	58
5.3	Simplex, Half-duplex and Full-Duplex Communication	58
5.4	Asynchronous and Synchronous Transmission	58
5.5	Multiplexing Signals	60
5.5.1	Synchronous Time Division Multiplexing (TDM)	61
5.5.2	Statistical Time Division Multiplexing (STDM)	62
5.5.3	Wavelength Division Multiplexing (WDM or FDM)	63
5.5.4	Combining WDM and STDM	63
5.6	Digital Subscriber Line (DSL)	64
5.7	Switching Techniques	66
5.7.1	Circuit Switching	66
5.7.2	Packet Switching	67
5.7.3	Cell Switching	68
5.8	Exercises	68
6	Physical Data Communication Interfaces, Standards and Errors	69
6.1	Objectives of this Chapter	69
6.2	Electrical Signal Interfaces	70
6.2.1	Serial Analog Interface	70
6.2.2	Serial Digital Interface	72
6.3	Optic Signal Interfaces	75
6.3.1	Synchronisation and SONET/SDH	77
6.3.2	The relationship between SONET and SDH	79
6.4	Transmission Errors	80
6.4.1	Sources of Transmission Errors	80
6.4.2	Error Rates	80
6.4.3	Error Codes	81
6.4.4	Hamming Distance	81
6.4.5	Parity	82
6.4.6	Error-Correcting Codes	83
6.4.7	Block check characters	84
6.4.8	Cyclic Redundancy Coding	85
6.5	Exercises	89

7	Data Link Protocols	91
7.1	Objectives of this Chapter	91
7.2	Principles of Protocol Design	93
7.2.1	Acknowledgements	94
7.2.2	Flow Control	94
7.2.3	Sequence Numbers	95
7.2.4	Piggybacking	96
7.2.5	Sliding Window Protocols	97
7.2.6	Pipelining	99
7.2.7	Positive Acknowledgements	101
7.3	Synchronous Data Link Control (SDLC)	101
7.3.1	Flag field	102
7.3.2	Address field	103
7.3.3	Control field	104
7.3.4	Acknowledgement and Retransmission.	105
7.3.5	Supervisory Format	106
7.3.6	Nonsequenced format	107
7.4	The Data Link Layer on the Internet	107
7.4.1	Serial Line IP (SLIP)	107
7.4.2	Point-to-Point Protocol (PPP)	108
7.5	Exercises	109
III	Network Protocols	111
8	Local Area Networks	113
8.1	Objectives of this Chapter	113
8.2	Introduction	114
8.3	Classification of LANs	115
8.3.1	Topologies	115
8.3.2	Common Bus Topology	115
8.3.3	Star Topology	116
8.3.4	Ring Topology	116
8.3.5	Media Access Control	116
8.4	Random Access Techniques	117

8.4.1	1-persistent CSMA	118
8.4.2	Non-persistent CSMA	118
8.4.3	CSMA/CD	118
8.4.4	CSMA/CA	119
8.4.5	IEEE LAN Classification	119
8.5	Logical Link Control (IEEE 802.2)	119
8.6	Ethernet	121
8.6.1	Ethernet Performance	125
8.6.2	Fast Ethernet	126
8.7	Deterministic or Token Passing Protocols	127
8.7.1	Token Ring (IEEE 802.5)	127
8.7.2	Ring Maintenance	129
8.7.3	Token Ring Performance	130
8.7.4	Token Passing Bus (IEEE 802.4)	131
8.8	Wireless LAN (IEEE 802.11)	133
8.9	Network Interconnection	134
8.10	Exercises	135
9	Wide Area Networks	137
9.1	Objectives of this Chapter	137
9.2	Overview	138
9.3	Datagram Service, Virtual Circuit Service	138
9.4	Internal Structure of the Network	140
9.5	Routing Techniques	143
9.6	Congestion Control	145
9.6.1	Preallocation of Buffers	146
9.6.2	Discarding Packets	146
9.6.3	Choke Packets	147
9.7	Flow control	148
9.8	Deadlock Prevention	149
9.9	Frame Relay Networks	150
9.9.1	Circuit Connections	152
9.9.2	Committed Information Rates	152
9.9.3	Local Managment Interface (LMI)	154

9.9.4	Frame Relay Applications	154
9.10	The X.25 Packet Interface	155
9.11	Internet Network Layer Protocol (IP)	161
9.12	Asynchronous Transfer Mode (ATM) Networks	163
9.12.1	Quality of Service (QoS)	165
9.12.2	ATM Cells	166
9.12.3	ATM Service Classes	168
9.12.4	Real-Time Services	168
9.13	Exercises	170
10	Transport Layer	173
10.1	Objectives of this Chapter	173
10.2	Introduction	174
10.3	ISO Transport Protocol	174
10.3.1	Establishing a connection	175
10.3.2	Flow control in the transport layer	177
10.4	Internet Transport protocols	178
10.4.1	User Datagram Protocol (UDP)	179
10.4.2	Transmission Control Protocol (TCP)	180
10.5	UNIX Sockets	182
10.6	Asynchronous Transmission Mode (ATM)	183
10.6.1	Generics of the ATM Adaptation Layer	184
10.6.2	AAL 1	184
10.6.3	AAL 2	186
10.6.4	AAL 3/4	186
10.6.5	AAL 5	188
10.7	Exercises	189

List of Figures

1.1	Mainframe computer around 1970	6
1.2	One typical configuration of a terminal-oriented network.	7
1.3	A typical configuration of a terminal-oriented network using Statistical Multiplexers.	8
1.4	A typical configuration of a terminal-oriented network using a FEP.	8
1.5	Examples of analog and digital signals	10
1.6	Manchester encoding of the byte 1111 0000.	11
1.7	The ISO OSI Reference Model.	13
1.8	Layers, protocols, and interfaces.	15
1.9	TCP/IP protocol suite	18
2.1	Schematic of a conducting coil rotating at speed ω radians per second through a field of magnetic flux, generating a current in the conductor as it does so	24
2.2	Illustrating phase differences in a sinusoidal signal	25
2.3	Shielded twisted-pair line	27
2.4	A typical coaxial cable.	28
2.5	Sources of attenuation and distortion of a digital signal	29
2.6	A binary signal and its Fourier components.	32
2.7	Two binary signals: $g_2(t)$ with double the <i>capacity</i> (number of bits per second) of $g_1(t)$. . .	33
2.8	Illustrating the effect of bandwidth on a digital signal	35
2.9	Some digital encoding techniques	36
2.10	A binary signal (a) amplitude modulated (PAM) (b) frequency modulated (MFSK) (c) phase modulated (PSK)	38
2.11	Illustrating the principles of Pulse Code Modulation (PCM)	39
2.12	A transmitted byte	41
3.1	The Optic Telegraph	44
3.2	Principles and components of optic fibres	45
3.3	(a) Step index optic fibre	47

3.4	(b) Multimode fibre	47
3.5	(c) Single mode fibre	47
3.6	(d) Graded index fibre	47
4.1	Satellite communication	55
5.1	Simplex, Half-duplex and Full-duplex Communication	58
5.2	Asynchronous and Synchronous Transmissions	59
5.3	Multiplexing	61
5.4	Synchronous Time Division Multiplexing	61
5.5	Statistical Time Division Multiplexing	62
5.6	Wavelength Division Multiplexing	63
5.7	Asymmetric Digital Subscriber Line (ADSL) technology using FDM only	64
5.8	Asymmetric Digital Subscriber Line (ADSL) technology using Echo-suppression	65
5.9	Circuit switching	66
5.10	Conceptual view of packet switching	67
6.1	RS-232C/V.24 interface	70
6.2	RS-232C Connector	71
6.3	Sending and receiving over an RS-232 connection	71
6.4	Signal lines used in the X21 serial digital interface	73
6.5	X21 protocol state transition graph	74
6.6	SONET configuration.	76
6.7	The STS-1 SONET frame.	78
6.8	Mapping basic signals onto an STS-N channel.	79
6.9	Error burst examples.	81
6.10	Example of block sum parity checking	85
7.1	Conceptual model of Layer 2, 3 and 4 protocols	93
7.2	The Alternating Bit (AB) protocol	96
7.3	A sliding window of size 1, with a 3-bit sequence number. Part (a) initially (b) after the first frame has been sent (c) after the first frame has been received (d) after the first acknowledgement has been received – page 85	97
7.4	Pipelined ARQ protocols	100
7.5	The SDLC frame format.	102
7.6	An example of bit stuffing	103

7.7	Three formats for the basic SDLC control field	105
7.8	The PPP full frame format for unnumbered mode operation	109
8.1	LAN Bus topology	115
8.2	LAN Star topology	116
8.3	Ring topology.	117
8.4	Correspondence between IEEE 802 layers and the OSI model	120
8.5	(a) Position of LLC and (b) Protocol formats	120
8.6	Ethernet 10BASE5 topology	122
8.7	CSMA/CD frame format	123
8.8	CSMA/CD MAC process	123
8.9	Popular networking configuration	124
8.10	Ethernet states: contention, transmission, or idle.	125
8.11	Token-Ring frame format	127
8.12	Token-Ring medium access algorithm	128
8.13	Token-Ring frame format and content	129
8.14	Token bus network	132
8.15	Wireless LAN communication	134
9.1	One directional Virtual Circuits showing the order on which they are set up	141
9.2	NODE tables corresponding to the Virtual Circuits set up Fig. 9.1	142
9.3	Congestion in the Service Provider network.	145
9.4	Congestion in a NODE can be reduced by putting an upper bound on the number of buffers queued on an output line	147
9.5	Store-and-forward lockup: (a) direct, (b) indirect.	149
9.6	Message re-assembly lockup.	150
9.7	Packet Level (top) and Frame Relay (bottom) PDU formats	151
9.8	Packet and Frame Relay processing flow diagrams.	153
9.9	Illustrating Committed Information Rate (CIR) and discard eligibility of frames	154
9.10	The DTE-DCE X.25 ITU-T interface standard for Wide Area packet switched networks.	155
9.11	X.25 CALL REQUEST packet format.	156
9.12	Sequence diagram of X.25 call establishment, data transfer and call clearing phases	157
9.13	X.25 Control packet format	158
9.14	X.25 data packet format	160
9.15	The IP (Internet Protocol) header format	161

9.16 Source and destination address formats in the IP header	163
9.17 The ATM principle	164
9.18 ATM connection paths	165
9.19 ATM call establishment	166
9.20 ATM cell format at (a) User-Service Provider interface and (b) internal to the network	167
9.21 Virtual circuit configuration.	171
10.1 Establishing a connection using a three-way handshake	177
10.2 Flow control mechanism	178
10.3 Internet transport protocols TCP and UDP in context	179
10.4 TCP segment layout	180
10.5 The ATM protocol stack	183
10.6 The AAL 1 cell format	185
10.7 AAL 2 cell format	186
10.8 AAL 3/4 convergence sub-layer message format	187
10.9 AAL 3/4 cell format	187
10.10 AAL 5 convergence sub-layer message format	188

List of Tables

4.1	Electro magnetic spectrum and its uses	53
6.1	Typical payloads.	76
6.2	Transmission speeds of SONET and SDH.	79
6.3	The Hamming code explained	83
6.4	The Hamming code bit positions	83
6.5	Calculation of the polynomial code checksum.	87
8.1	IEEE 802.3 networks with 512 bit slot times, 96 bit gap times and jam signals lasting 32 – 48 bits	121
8.2	Token ring control frames.	130
9.1	Summary of the major differences between Virtual Circuit and Datagram service.	140
9.2	LAN and WAN characteristics compared	155
9.3	X.25 packet format	156
9.4	Values of the Pay Load type (PT) field in an ATM header cell	168
10.1	TCP user primitives and their function	181
10.2	Service classes supported by AAL	184

Part I

Overview of Telecommunication and Computer Networks

Chapter 1

Basic Concepts

1.1 Objectives of this Chapter

There is a great deal to know about telephone and computer networks, or more correctly *telecommunication* networks since with the advent of the digital age and cellular networks, the distinction is getting increasingly fuzzy.

1. We start off by explaining the origins of computer networks and why the telephone network has always played a role in computer networks from the very start.
2. Without understanding the difference between analog and digital signals, one will never comprehend why networks are playing such an increasingly important role in the world of computing, so we explain those fundamentals. In some countries of the world, this is the sort of information that is taught in the 13th grade at school. Maybe it is better than learning Shakespeare, but I doubt it.
3. Computers, like humans need rules when communicating. These are called protocols as we shall explain.
4. The software that makes communication between computers possible is some of the most complicated that can be written. In order to understand how it works, we divide it into layers which group the functions; hence network architecture.
5. Since computers communicate across continents and national boundaries, we need international organisations to make and approve the rules or protocols. There are many as we describe in the last part of this chapter.

A reader who understands everything discussed in this chapter will be very aware of how much there still is to learn in telecommunication and computer networks, but will have a clear understanding of the very fundamental concepts.

1.2 Introduction

A decade or two we associated the word “communication” with the use of the telephone or radio. While those applications remain we nowadays think almost exclusively of the Internet and probably wireless, cellular telephones. It has become so pervasive that we simply take it for granted. One of the greatest impacts that networking has had on society is to bring together vast sources of information, be they human minds or computer systems: the emergence of a “*global village*” is changing the way society interacts and does business.

An understanding of the principles behind computer networks is therefore something that every student should have. Communication and Information Technology is advancing at an exponential rate, and we are living in an exciting age where we are shaping the future of this technology. This introduction seeks to provide the student with the core concepts behind communication networks.

1.3 Evolution of Computer Networks

The evolution of computer networks is best traced by following the development of computing resources used by large organisations. The earliest commercially available computers were characterized by expensive hardware and relatively primitive software. Typically, an organisation would purchase a single computer which would then be centrally located in a large air-conditioned room. It would consist of a central processing unit with a limited quantity of primary memory, some secondary disc storage, a printer, a punch-card reader and an operator console. Something looking somewhat like that in Fig. 1.1.

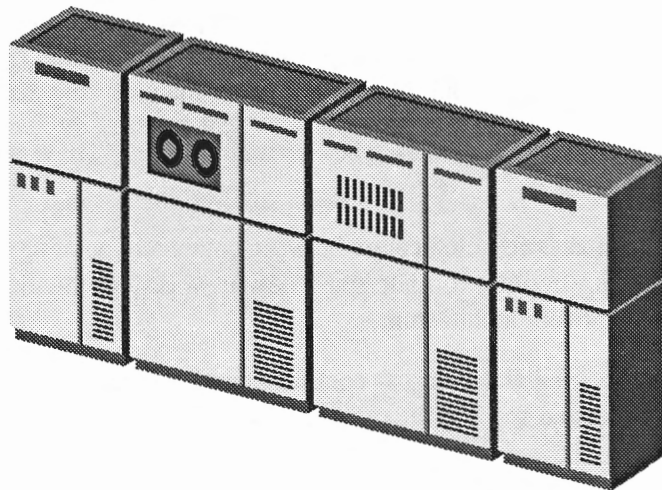


Figure 1.1: Mainframe computer around 1970

Users normally prepared their data *off-line* on a card punch located in a different room, and the computer operator would then load and run the prepared programs sequentially.

As computer hardware became faster and its operating software advanced, fast secondary storage – large magnetic drums and later magnetic disks – and multi-programming operating systems came into existence. This made it possible to *timeshare* the central processing unit between a number of active programs, thereby

allowing several users to run their programs *interactively* and to access stored data simultaneously via their own individual terminal. The terminals were normally electromechanical teletypewriters (TTYs) similar to those already in use at the time in international telex networks. They were designed, therefore, to transmit and receive one character at a time over long distances in *serial (or sequential) mode*.

The computers then became known as *multi-access systems* providing *on-line* access to stored programs and data. Initially, the terminals were all located in close proximity to the main computer complex with a simple serial line connection between the computer and the terminal.

Users increasingly found it useful though to have a computer terminal on their desk. Within a building or on a campus that posed no problem. However, over longer distances the expense became important and, above all, communication (of voice at first, but now data as well) was the monopoly of a central public organisation: Telkom!

Technically there was the additional issue that telephone networks carried analogue signals (explained in the next section) and computers used binary or digital signals. Hence modems¹ were invented to carry computer communications nationally over wide geographical areas. An operational computer system typical at that time is shown in Fig. 1.2 where the letters *PSTN* stand for *Public Switched Telephone Network*.

CENTRAL COMPUTER

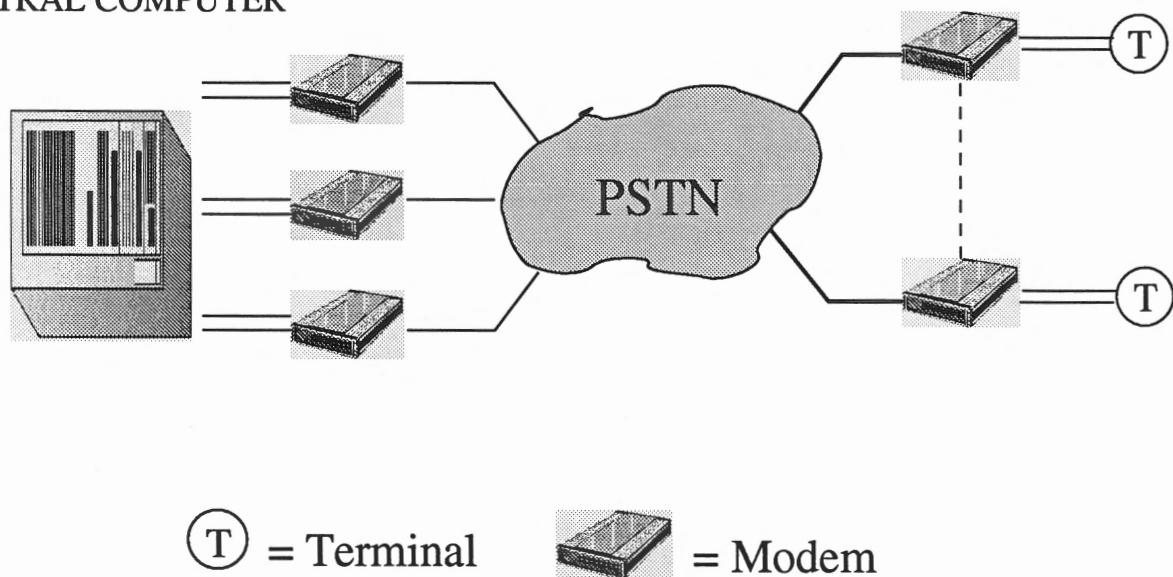


Figure 1.2: One typical configuration of a terminal-oriented network.

The use of a switched (“switched” in that connections could be changed from time to time to connect two individual users, the computer and the terminal in this instance) telephone network as the basic data communication medium meant that the communication line costs could no longer be regarded as insignificant and, indeed, soon constituted a substantial proportion of the system operating costs. To minimize these costs, therefore, devices such as *statistical multiplexers* and *cluster controllers* were introduced. Essentially, these allowed a single communication line, often on permanent lease from the telecommunications authorities, to be shared between a number of simultaneous users all located, for example, at the same remote site. An example of such a system is shown diagrammatically in Fig. 1.3.

In addition, the increasing level of usage of computers soon gave rise to systems of hundreds of computers

¹ a modem or *modulator-demodulator* is used to convert the digital signals of a computer into equivalent analogue signals. These concepts are explored later in this chapter

CENTRAL COMPUTER

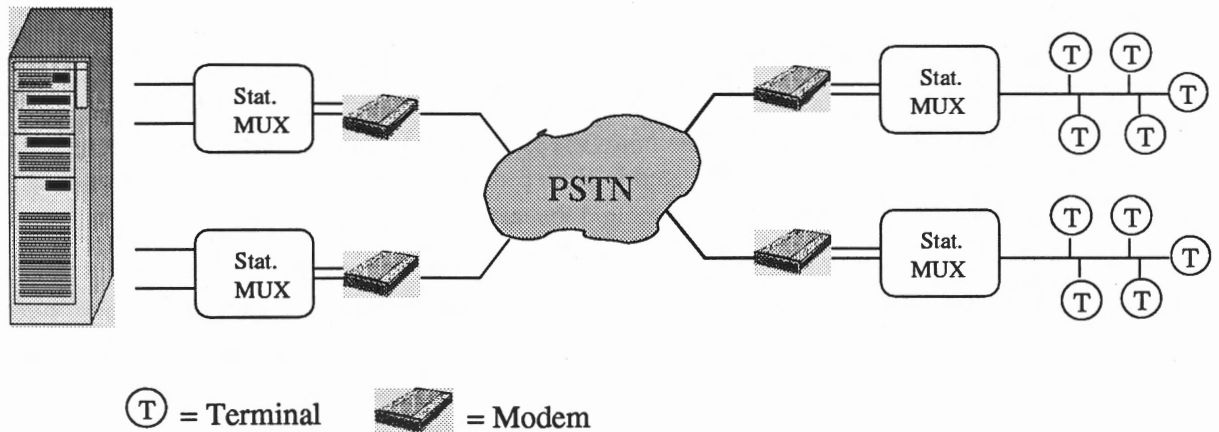


Figure 1.3: A typical configuration of a terminal-oriented network using Statistical Multiplexers.

within an organisation. The effect of this was that the central computer could no longer cope with the processing over heads associated with servicing the various communication lines in addition to its normal processing functions. Consequently a special device, called a *Front End Processor (FEP)* was introduced for the job of controlling the communication lines. Such a system is shown diagrammatically in Fig. 1.4.

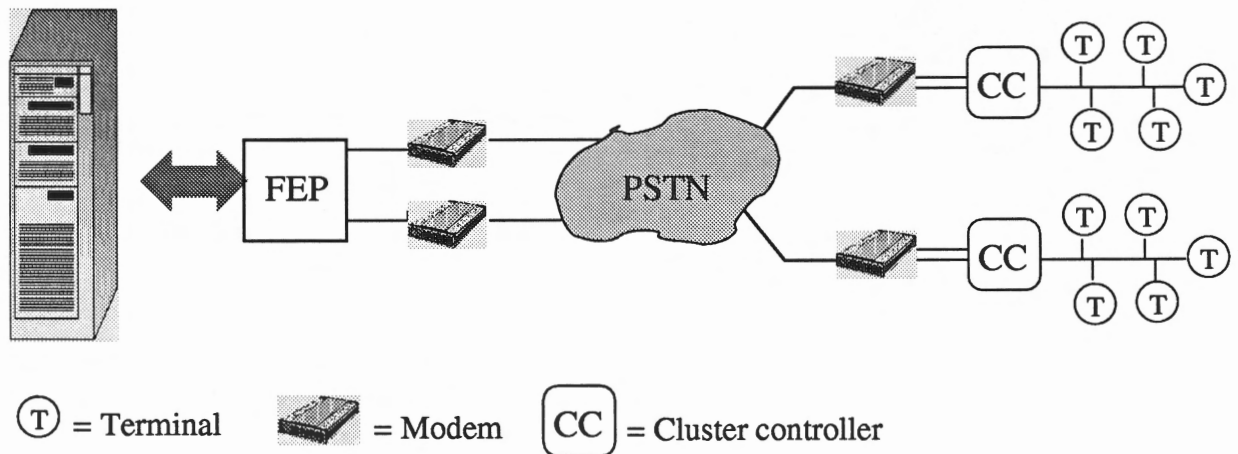


Figure 1.4: A typical configuration of a terminal-oriented network using a FEP.

The structures shown in Fig's 1.3 and 1.4 were particularly prevalent in organizations normally holding large amounts of data at a central site, such as the major clearing banks and airlines.

In many other organisations, it is not necessary and in fact desirable not to hold all the data centrally, and hence it became common place for organisations to have a number of autonomous computer systems located at different geographical locations. Typically, these systems provided a local computing function, but there was often a requirement for them to communicate with each other to share resources and data. In such a network, the internal message units used may be long. This can result in a significant delay between a message entering the network and the time it leaves the network (known as the *response time*), due to long messages waiting upon each other at the various resources along the way.

It then became economically feasible for TelCo's² to provide a separate, autonomous *computer communication network*. Such communication networks that carry mostly computer data normally operate using smaller units of data known as a *data packet*. It is therefore commonly referred to as a *Packet Switched Network*. Increasingly the term "*IP network*" is also used for reasons that will become clear only much later. Also, since the interconnected computers are normally geographically widespread, such a network is also called a *Wide Area Network (WAN)*.

The next obvious development was that computers of different manufacturers (IBM, SUN Microsystems, Microsoft, etc) with their different architectures and applications need to communicate with one another. The need for open (for everybody to see) interface standards became all important. This was the status at the beginning of the 1980's since being overtaken by the event of the Internet in the middle of the last decade.

What was to become the Internet, was first proposed in 1967 in the United States of America as an experimental computer network. For a number of years before 1967, DARPA, the (Defense) Advanced Research Projects Agency of the United States Department of Defense had been funding the growth and development of many multiclass timeshared computer systems at a number of university and industrial research centres across the United States. By 1967, many of these had shown themselves to be valuable computing resources and it was recognized that the Department of Defense as well as the scientific community could benefit greatly if there were to be made available a communication service providing remote access from any terminal to all of these systems. In early 1969 a contract was awarded for the implementation of the ARPANET to Bolt, Beranek and Newman (BBN) an engineering firm based in Massachusetts.

Simultaneously DARPA funded the research which resulted in a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic. These are now known as the *Transmission Control Protocol/Internet Protocol* or TCP/IP which can be used to communicate across networks which represent a wide variety of underlying network technologies. What started as ARPANET is now known as the Internet, literally a network that (Inter-)connects many worldwide networks. The evolution has however not ended and is being fueled further by the popularity of wireless, cellular communication. In order for the Internet to reach the very edge of the network the last bastion of traditional voice transmission has to be broken: In other words, transmitting voice over data (or "IP") networks, including the wireless link.

Now that we now a little about their evolution, we shall start from scratch to understand the exciting world of telecommunication networks.

1.4 Electrical Signals

Signals to represent data, voice conversations or video transmissions are either electromagnetic radio waves, electrical signals flowing in a conductor or optical (light) signals. Optical signals are being used more and more but we shall discuss those later.

An electrical signal is transmitted along a conductor by varying some electrical property such as voltage or current which could either be an *analog* or *digital* signal as illustrated in Figure 1.5.

Electrical signals can be either *analog* or *digital* as illustrated in Fig. 1.5. Digital signals are also called binary signals. An analog signal is continuous in time whereas digital signals take on discrete values and changes instantaneously from one voltage or current value to the other in time. The number of signal levels are always 2^k for $k = 1, 2, \dots, n$. Originally $k = 2$ but as technology improves it is normal for more than

² A general term for the Telecommunication Company Monopolies which alas exist in too many countries still.

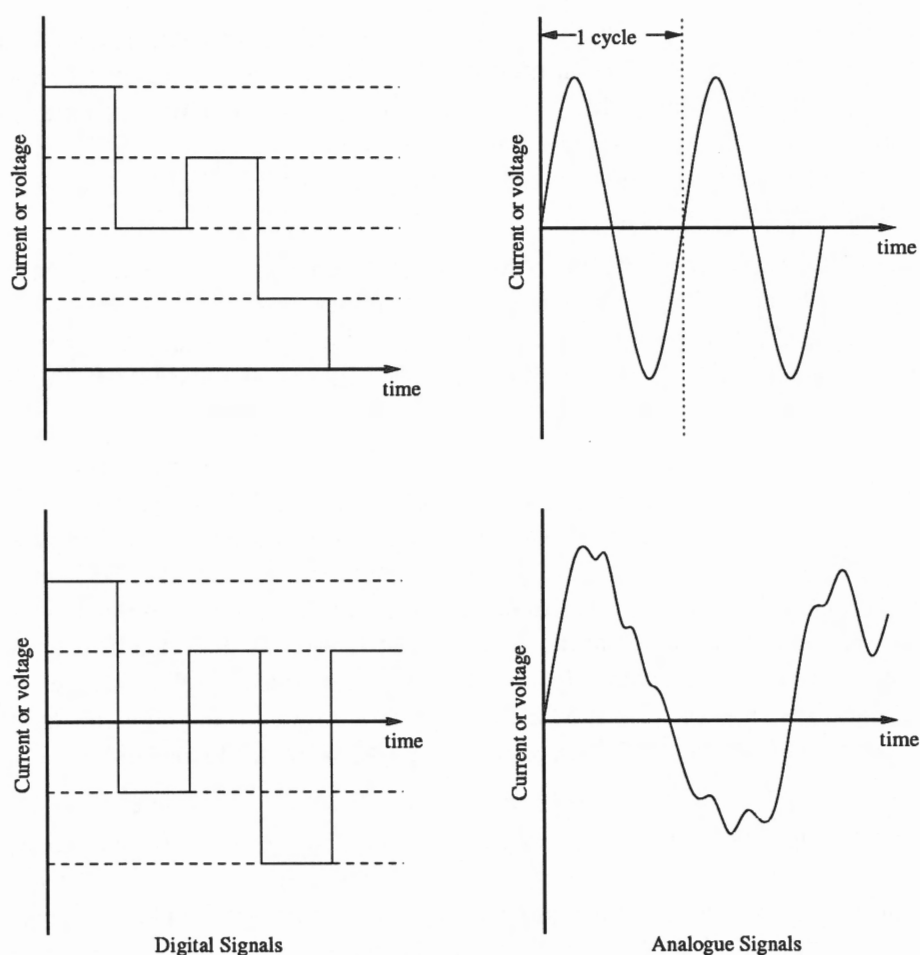


Figure 1.5: Examples of analog and digital signals

two levels to be represented. In practice the way in which the binary data are represented or “coded” by the digital signal is all but simple. One method commonly used is known as *Manchester Encoding* illustrated in Figure 1.6.

Since analog signals vary continuously in time they can be represented by mathematical formulae and manipulated accordingly. Analogue signals do not have to be cyclical in that they repeat themselves after a fixed time T called the period. If they are, the number of complete cycles per second (also called “Hertz”) is the inverse of the period and is called the **frequency**, usually denoted by the symbol f . Thus $f = 1/T$ cycles/second, or $T = 1/f$ seconds. The signals with periodic wave forms are the most useful to us in telecommunication since can *modulate* them to carry digital signals as we shall see in Chapter 2.

Analog signals not only represent current flowing in a copper conductor, but most importantly may be electro magnetic waves travelling through space in which case we call them *radio waves*. Depending on their frequency, radio waves can travel around the world, which means that everyone can receive them. Although this is exactly what radio waves were originally intended for, it is simultaneously a problem if too many parties want to use the limited frequency spectrum.

When the frequency of the electro magnetic waves is very high we refer to them as *microwaves* which have the “advantage” (for the intended purpose) that they are easily attenuated by obstacles like buildings in their way (the same effect that chickens in a microwave experience) and are only really reliable when they are

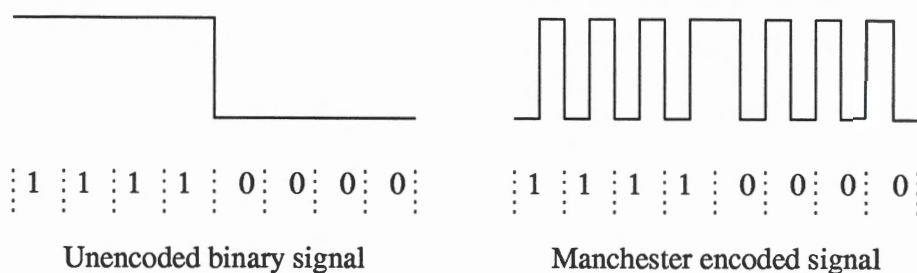


Figure 1.6: Manchester encoding of the byte 1111 0000.

transmitted to a receiver which is in direct line of sight. With this property it is possible to “re-use” the scarce frequencies in geographically separated areas. If we transmit the microwave signals at low power in addition, we can divide a geographical area into “cells” where the transmissions do not interfere with each other provided the frequencies in adjacent cells are different. This is the way that a GSM (cell phone) network works. We shall return to all these concepts in later chapters.

1.5 Optical Signals

Telecommunication Networks also use optical fibre to transport data encoded as ON/OFF pulses of light. Depending on the type of fibre, a transmitter at one end of the fibre uses a *light emitting diode (LED)* or *Laser* to send pulses of light down the fibre. A receiver at the other end uses a light detector such as a *Photo Intrinsic Diode (PIN)* or *Avalanche Photo Diode (APD)* to detect the pulses.

The number of bits per second (Bps) that can be transmitted along an optical fibre is virtually unlimited since it is theoretically possible to send several beams (called *modes*) of light of different frequencies down a fibre, each carrying typically 2 GBps³. We will return to optical signals and the associated standards in Chapter ??.

1.6 A Matter of Protocol

Now that we understand a little about the signals travelling between two communicating parties in a telecommunications network, we turn to the rules of communication or *protocols* which are needed for the parties to exchange information without error.

Essentially, a computer network consists of one or more computer “talking” with each other. The *rules that govern the exchange of information*, and the *exact structure of that information* are known as a *protocol*. As an introduction to the idea of a protocol, we consider an analogy from everyday life namely, a telephone conversation between two people living in different cities in different countries. As we all know the process is something like the following:

First of all, the caller picks up the telephone receiver. If a dialing tone is heard, he will select the number of the callee on the buttons of his telephone. The *called number* constitutes the *address* of the callee. If no dialing tone is heard, the caller would normally discontinue his attempts to place the call.

Once the call goes through, the remote instrument either rings and is answered or not. The callee could also be using the device. Should the callee answer the ringing telephone, a “verbal handshake” typically occurs. Note that in this case it is a *name*, rather than a *number*, that identifies the caller and callee to each other.

³A Gigabit is 10^{12} bits

Exam: Definition of Protocol: Format and content of the messages and the rules for exchanging them.

Now that the connection has been established, the two parties conduct a conversation. Certain conventions also apply in this case. Two people do not (normally that is!) talk simultaneously; if they did, a process of “re-synchronization” would need to occur. If one person speaks too quickly, the other may ask him to slow down.

If the line should fade or if there is noise on the line, messages like “I did not hear your last words” or “Would you please repeat the last sentence?” would ensure that the information passing between the two parties remains error-free.

Should an abrupt cut-off occur, we know from experience that a frustrating situation of “deadlock” might occur, when both people try simultaneously to call each other. There are no rules in this case which is an “error” in the specification of the (human) “protocol”.

At the conclusion of their conversation the caller and callee go through similar conventions for terminating their call and would eventually put their respective telephone receivers down, thereby not only ending their conversation, but also breaking the physical (electrical) communication link.

When two processes in separate *computers* communicate with each other, similar conventions for:

- addressing
- call establishment and call termination
- error checking
- information recovery
- and flow control

must be decided upon.

Once again, the rules and conventions used in a session of communication are known as a protocol. The specification of a protocol also includes a specification of the format and content of the messages being passed. As computers are dumb, deterministic machines, they require a far more elaborate and exact protocol than in the case of a conversation conducted in natural language between two human beings.

The telephone analogy also reveals another basic concept important in the study of computer communication. This is the fact that there are a number of stages to the “conversation” between communicating computer processes:

- call-establishment phase
- data (information) transfer phase
- call-clearing phase
- idle phase

1.7 Network Architecture

Computers have different architectures that use different languages, store data in different formats, and communicate at different rates. Consequently there is much incompatibility, and communication is difficult. In fact, *how do computers manage to communicate at all ?* They communicate the same way that the

people in our analogy do. In the analogy, there are rules which apply at the “physical level” (number called, dialling tone, engaged signal, etc) and rules which apply at, shall we say “logical” level between the humans themselves. The same is true for computers.

As long ago as 1976 the International Standards Organisation (ISO) recognised the need for some standardisation of computer network interfaces. It was apparent even at that early stage, that for the reasons mentioned already, it would be expedient to describe the architecture of the network software in layers in order to facilitate the interconnection of unlike systems. In other words, to provide an *open system* that any other system could connect to. ISO thus set out to standardise the *Open System Interconnection Reference Model*, or OSI reference model for short, intended as a reference architecture for network design. This model is illustrated in Fig. 1.7.

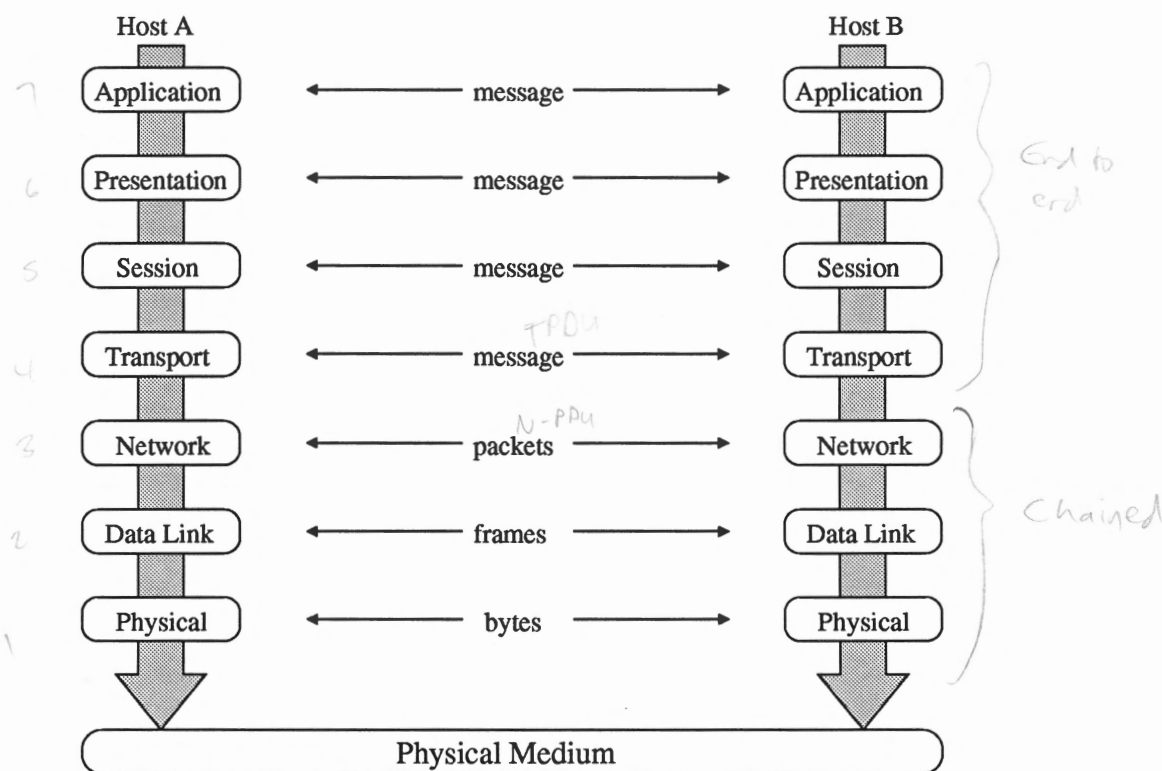


Figure 1.7: The ISO OSI Reference Model.

Between each pair of adjacent layers in the OSI model there is an interface. The interface defines which *services* the lower layer offers to the upper one. When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is having clearly defined interfaces between the layers. Having clearly defined interfaces, in turn, requires that each layer performs a *specific collection of well understood functions*. In addition to minimizing the amount of information that must be passed between layers, clear cut interfaces also make it simpler to replace the implementation of one layer with a completely different one (e.g., when all the telephone lines are replaced by satellite channels), because all that is required of the new implementation is that it offers exactly the same set of services to the next layer higher up as the old implementation did.

The set of layers and protocols is called the *network architecture*. The specification of the architecture must contain enough information to allow an implementor to write the program for each layer so that the program will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification


of the interfaces are part of the architecture.

When describing the operation of any of the protocol layers, it is important from the outset to discriminate between:

1. the **services provided** by the layer
2. the **protocol** (i.e., logical operation of the layer)
3. the **services used** by that layer

This is important because only then can the function of each layer be defined in the context of the other layers. This also has the effects that a person implementing one protocol layer needs only to have a knowledge of the services the layer is to provide to the layer above, the internal operation (protocol) of the layer, and the services that are provided by the layer below to transfer the appropriate items of information, called **Protocol Data Units** or **PDU**s associated with the protocol to the corresponding layer in a remote computer.

Equally, the specification of each protocol layer comprises two sets of documents:

1. *service definition document* describes services obtained from layer below, services offered to layer above
2. *protocol specification document* specifies how two peer layers communicate 
state transition diagram/table.

The service definition document contains a specification of the services *provided by that layer* to the layer above it – that is the *user services*. These are in the form of a defined set of *service primitives*. These service primitives are similar to procedure calls in a high level language, each with an associated set of *service parameters*. These primitives are invoked to and from the layer through *service access points* (SAPs) (seen in Fig. 1.8), e.g. a *request*, *indication*, *response* and *confirm* primitive. As will be seen in later chapters, it is through the service primitives that the layer above achieves the transfer of information to the correspondent layer in a remote system.

We need to clarify the rather cryptic notations N , $N+1$ and $N-1$, seen in Fig. 1.8. A layer in the OSI model is generally referred to as the (N) -layer. The layer above it (if one exists) is designated as the $(N+1)$ -layer, similarly the one below it is the $(N-1)$ -layer. The peer process abstraction is crucial to all network design. Without this technique, it would be impossible to partition the design of the complete network into several smaller, manageable, design problems, namely the design of the individual layers.

It was not intended that there should be a single standard protocol associated with each layer. Rather a set of standards is associated with each layer, each offering different levels of functionality. In each of the following sections we briefly discuss each layer of the architecture

1.7.1 Physical layer

The physical layer is concerned with *transmitting raw bits* over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit, it is received by the other side as a 1 bit, not as a 0 bit. Typical questions here are how many volts should be used to represent a 1 and how many for a 0, how many micro seconds a bit occupies, whether transmission is simplex or duplex, how the initial connection is established and how it is broken when both sides are finished, how many pins the network connector has and what each pin is used for. In some cases a transmission facility consists of multiple physical channels, in which case the physical layer can make them look like a single channel, although higher layers can also perform this function.

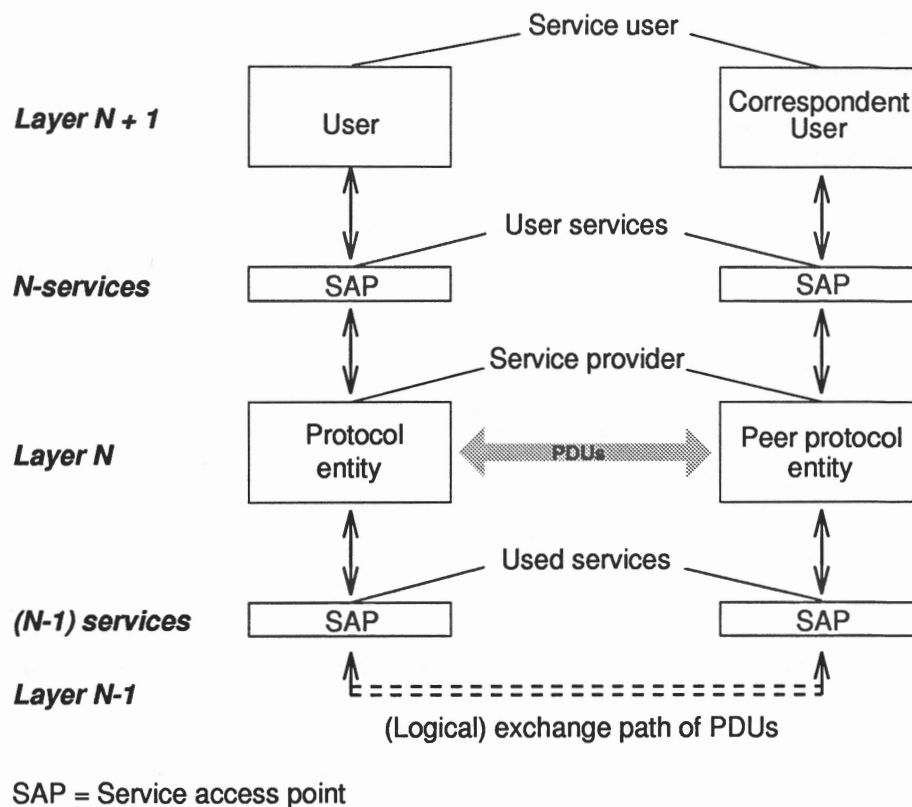


Figure 1.8: Layers, protocols, and interfaces.

1.7.2 Data link layer

The task of the data link layer is to take a raw transmission facility and transform it into a line that appears *free of transmission errors* to the network layer. It accomplishes this task by breaking the input data up into data frames⁴ transmitting the frames sequentially, and processing the acknowledgement frames sent back by the receiver. Since Layer 1 merely accepts and transmits a stream of bits without any regard to meaning or structure, it is up to the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame. These bit patterns can unintentionally occur in the data, so special care must be taken to avoid confusion. It is up to this layer to solve the problems caused by damaged, lost, and duplicate frames, so that network layer can assume it is working with an error-free (virtual) line.

1.7.3 Network layer

The network layer controls the operation of the subnet. Among other things, it determines the chief characteristics of the *DCE-DTE interface*⁵, and how *Network Protocol Data Units or NPDUs*, more commonly known as *packets*, the units of information exchanged in the network layer, are routed within the network. A major design issue here is the division of labour between the network node and host, in particular who should ensure that all packets are correctly received at their destinations, and in the proper order. What this

⁴Note that the ISO term for a "frame" is a "Data-Link-Service-Data-Unit". We shall use the term "frame".

⁵The interface between the Data Circuit terminating Equipment (DCE) or network node and the Data Terminal Equipment (DTE) or user equipment.

layer of software does, basically, is accept messages from the source host, convert them to packets, and see to it that the packets get directed towards the destination.

The network layer also determines what type of service to provide the transport layer. One type of service (virtual) point-to-point channel (called a *virtual circuit*) that delivers messages in the order in which they were sent. Another kind of network service, called *datagram service* gives no guarantee whether packets will be delivered or whether they will be delivered in sequence. The type of service is determined when the connection is established.

1.7.4 Transport layer

The basic function of the transport layer is to accept data from the session layer, split it up into smaller units, if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done in the most efficient possible way, and in a way that isolates the session layer from the inevitable changes in the hardware technology.

Under normal conditions, the transport layer creates a distinct network connection for each transport connection required by the session layer. The transport layer is a true source-to-destination or end-to-end layer. In other words, an application on the source machine carries on a conversation with another application on the destination machine, using the message headers and control messages. At the lower layers, the protocols are carried out by each machine and its immediate neighbours, and not by the ultimate source and destination machines, which may be separated by many DCEs. The difference between layers 1 – 3, which are chained, and layers 4 – 7, which are end-to-end, is also illustrated in Fig. 1.7.

1.7.5 Session layer

It is with this layer that an application process must negotiate to establish a connection with a process on another machine. Once the connection has been established, the session layer can manage the dialogue in an orderly manner, if the user has requested that service.

A connection between applications (or strictly speaking, between two session-layer processes) is usually called a *session*. A session might be used to allow a user to log into a remote time-sharing system or to transfer a file between two machines. To establish a session, the user must provide the remote address he wants to connect to. Session addresses are intended for use by users of their programs, whereas transport addresses are intended for the use by transport stations, so the session layer must be able to convert a session address to its transport address, to request that a transport connection be set up.

Setting up a session is a complicated operation. To start with, it may be necessary that each end of the session be properly authenticated, to prove that it has the right to engage in the session and to ensure that the correct party receives the bill later. Then the two ends must agree on a variety of options that may or may not be in effect for the session, such as whether communications is to be half-duplex or full-duplex.

Another function of the session layer is management of the session once it has been set up. For example, if transport connections are unreliable, the session layer may be required to attempt to transparently recover from broken transport connections. As another example, in many data base management systems, it is crucial that a complicated transaction against the data base never be aborted halfway, since doing so would leave the data base in an inconsistent state. The session layer often provides a facility by which a group of messages can be bracketed, so that none of them are delivered to the remote user until all of them have arrived. This mechanism ensures that a hardware or software failure within the subnet can never cause a transaction to be aborted halfway through. The session layer can also provide for ordering of messages when

the transport service does not. In short, the session layer takes the (possibly bare bones) communication service offered by the transport layer and adds application-oriented functions to it. In some networks, the session and transport layers are merged into a single layer, or the session layer is absent altogether, if all that the users want is raw communication service.

1.7.6 Presentation layer

The *presentation layer* is concerned with the presentation(syntax) of data transfer between two communicating application processes. This layer performs functions that are requested sufficiently often to warrant finding a general solution for them, rather than letting each user solve the problems. These functions can often be performed by library routines called by the user.

A typical example of a transformation service that can be performed here is text compression. Encryption to provide security is another possibility. Conversion between character codes, such as ASCII to EBCDIC, might often be useful.

Bg Erdian! Little Erdian, SSL

1.7.7 Application layer *does not contain the application*

The highest layer, the *application layer*, works directly with the user of application programs. The application layer provides user services such as electronic mail, file transfers and resource allocation. For example, the application layer on one end should appear to send a file directly to the application layer on the other end independent of the underlying network or computer architectures. These services are known as *application service elements* or *ASEs*. To discriminate between ASEs that perform common application-support services and those that perform specific application-support services, the term *common application service element* or *CASE* is used to refer to the first (e.g. File Transfer and Management (FTAM)), and the term *specific application service element* or *SASE* is used to refer to the second.

1.7.8 Standards and ISO

The ISO OSI Model was originally intended as a universal standard of course. Suffice to say that it is still a very useful conceptual model, but in practice it has been overtaken by the Internet protocols. The protocol architecture associated with the Internet is known as the Transmission Control Protocol/ Internet Protocol (TCP/IP). It includes both network-oriented protocols and application support protocols. Because TCP/IP is in widespread use with an existing internet, many of the TCP/IP protocols have been used as the basis for ISO standards. Fig. 1.9 shows some of the standards associated with the TCP/IP protocol suite.

1.8 International Standards Bodies

The coordination on a world-wide scale to ensure that people in one country can communicate with their counterparts in another is provided by an agency of the United Nations called the *International Telecommunications Union (ITU)*. ITU has three main organs, two of which deal primarily with international radio broadcasting and one of which is primarily concerned with telephone and data communication systems. ITU-T's task is to make technical recommendations about telephone, telegraph and, more recently, data communication network interfaces. These often become internationally recognised standards. One example is V.24 which specifies the placement and meaning of the various pins on the connector used by most asynchronous terminals and V.27 which specifies the electrical characteristics.

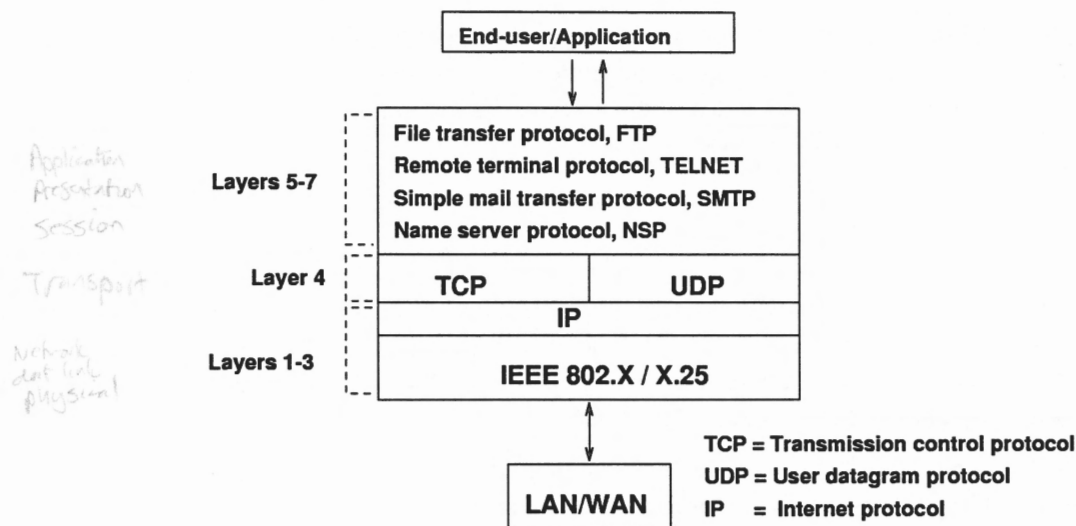


Figure 1.9: TCP/IP protocol suite

Other standard bodies in the United States are the *Institution of Electrical and Electronics Engineers (IEEE)* and in Europe ETSI (the European Telecommunications Standards Institute) is a non-profit making organization whose mission is to produce the telecommunications standards that will be used for decades to come throughout Europe and beyond. ETSI promotes the world-wide standardization process whenever possible. Its Work Programme is based on, and co-ordinated with, the activities of international standardization bodies, mainly the ITU-T and the ITU-R. There is also the *European Computer Manufacturers Association (ECMA)* and several other regional standards bodies on both sides of the Atlantic.

Not surprisingly there is a great deal of cooperation and consultation between the various bodies although each claims to be independent of the other. The ITU-T V.24 physical interface standard, for instance, is also known in the USA as the EIA standard RS232D.

The *International Standards Organization (ISO)*, is another group that tries to make the world a better place via standards. Its constituents are the national standards organizations in the member countries. On issues of telecommunication standards, ISO and ITU-T sometimes cooperate to avoid the irony of two official and mutually incompatible international standards.

1.9 Exercises

Exercise 1.1 Define a computer protocol.

Exercise 1.2 Explain why computers need a strictly defined communications protocol.

Exercise 1.3 A computer is sitting idle, waiting for an incoming call. Upon receiving this call, the two computers will exchange some data. Draw the state transition diagram for this machine, indicating the four phases of communication.

Exercise 1.4 Give two reasons for using a layered model.

Exercise 1.5 Which of the OSI layers would:

1. *break the transmission data up into frames?*
2. *decide the path that transmitted data will take?*
3. *provide secure data transmission?*
4. *do the conversion from ascii to EBCDIC?*

Exercise 1.6 *Why do you think it is important for the Data Link Layer to detect/correct errors?*

Exercise 1.7 *Does the OSI model specify a specific protocol for each layer? If it does, why does it? If not, why not?*

Exercise 1.8 *Regarding the OSI reference model:*

1. *Describe any layer.*
2. *Describe a typical service used by that layer.*
3. *Describe a typical service provided by the same layer.*
4. *How do you think the Service Access Points are implemented?*

Part II

Transmission Fundamentals

Chapter 2

Electrical Signals

2.1 Objectives of this Chapter

Although radio and optical signals are becoming very popular, the most common signals around are still electrical. In this chapter we introduce the basics one requires to understand the technologies to be described later.

1. We start by explaining that any signal can be composed into its Fourier components, which helps us to understand a lot about signal quality, capacity and bandwidth.
2. Electrical signals travel along conductors, usually copper, the various media are briefly explained.
3. We then investigate the properties of electrical channels and what their features and limitations are, including attenuation and the relationship between electrical noise and bandwidth.
4. The last sections in this chapter explain how one can transmit digital data as an analog signal and vice versa.

Without a firm grasp of the material in this chapter, the computer engineer or scientist will not follow much of the discussion about data synchronisation etc. which follow in subsequent chapters.

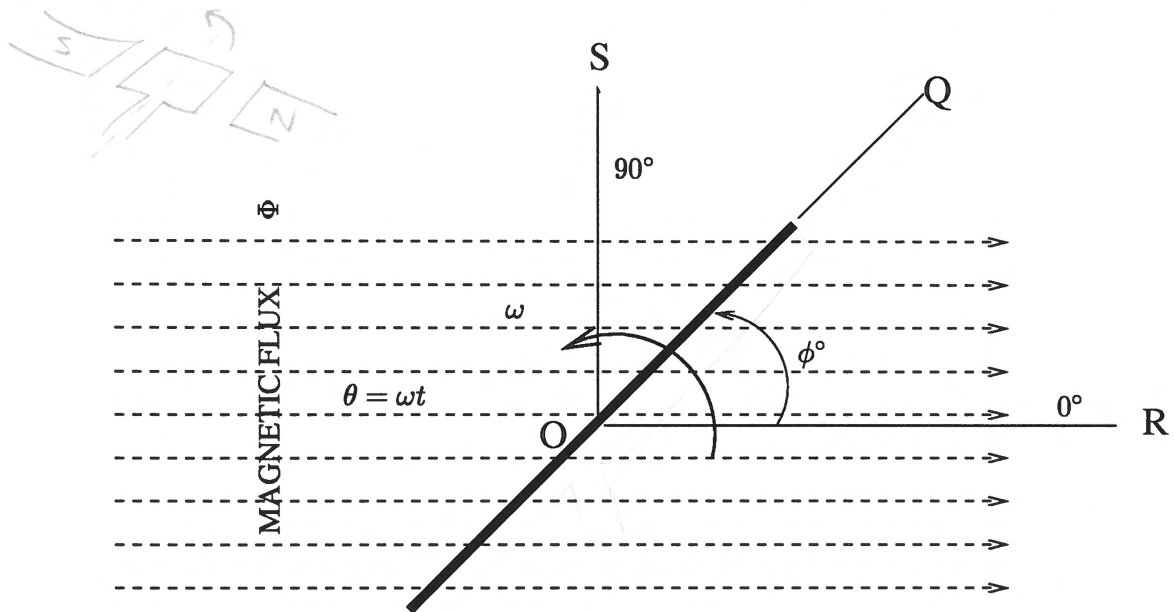


Figure 2.1: Schematic of a conducting coil rotating at speed ω radians per second through a field of magnetic flux, generating a current in the conductor as it does so

In order to learn about electrical signals and their properties we use the example of an electric current flowing in a conductor. Assume that the current is generated by a device known as an *alternator*, which is formed by rotating a coil of conductors through a field of magnetic flux as shown schematically in Fig. 2.1. Let the coil rotate at a constant speed ω radians/sec about an axis (at O in the figure) which is perpendicular to the uniform magnetic field illustrated by the flux lines. If time is measured from the instant that the coil is horizontal in position OR the amount of flux passing through the coil is given by $\Phi = \Phi_m \cos \theta$ or $\Phi = \Phi_m \cos \omega t$ where $\omega t = \theta$.

The voltage ¹ v induced in the coil, is given by the Faraday-Lenz Law and equals $-d\Phi/dt$, so that

$$(2.1) \quad -\frac{d\Phi}{dt} = v = \omega \Phi_m \sin \omega t$$

$$(2.2) \quad = V \sin \omega t$$

where $V = \omega \Phi_m$.

If, instead of measuring time from the instant when the coil is in position OR in Fig. 2.1, it is measured from the point where the coil is at an angle ϕ_1 from the horizontal (position R_1 in Fig. 2.2), then Eq. 2.1 becomes

$$v = V \sin(\omega t + \phi_1).$$

Similarly if we should start at position R_2 we have

$$v = V \sin(\omega t - \phi_2).$$

Note that the various signals in Fig. 2.2 are designated $i = I \sin(\omega t + \phi_1)$ and so on by virtue of Ohm's law from which $i = \frac{V}{Z} \sin(\omega t + \phi_1)$.

The angle part $(\omega t + \phi_1)$ of, say, $v = V \sin(\omega t + \phi_1)$ expression is called the *phase* at time t . In fact, the only difference between the two expressions is in the phases. The angle ϕ_1 is this phase difference and is

¹ Actually, electromagnetic force or e.m.f.

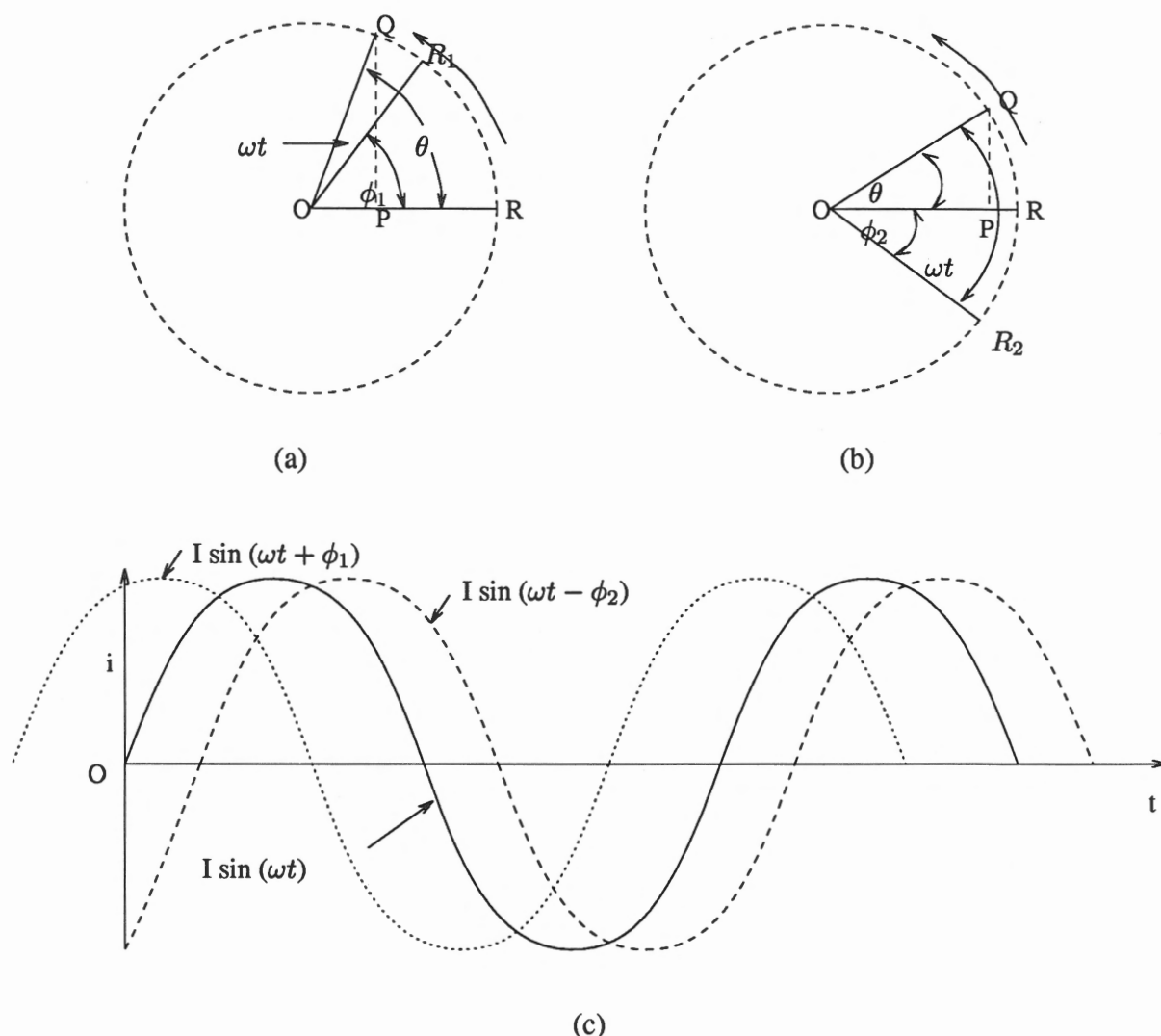


Figure 2.2: Illustrating phase differences in a sinusoidal signal

called the *phase angle*. At time 0 we have $v = V \sin \phi_1$ and Q is $\phi_1/2\pi$ of a cycle beyond R, so that a positive phase difference means a lead in phase.

Similarly, if $v = V \sin(\omega t - \phi_2)$ and the phase angle is now a lagging one when compared with the original one. The corresponding waveforms for the three cases are shown in Fig. 2.2(c). The effects of having leading and lagging phase angles can clearly be seen.

Note that an analog signal such as $v = V \sin(\omega t + \phi_1)$ has three independent parameters namely amplitude V , frequency f ($\omega = 2\pi f$ as we know) and finally phase ϕ_1 . If two or more sinusoidal alternating quantities with identical frequencies are added, their resultant is also a sinusoidal quantity, having the same frequency as that of the components, as can easily be verified.

2.2 Fourier Analysis

One can prove that any reasonably behaved periodic function, $g(t)$ with period T , can be constructed by a possible infinite, sum of sine and cosine functions:

$$(2.3) \quad g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

↑
harmonic number

where $f = 1/T$ is the fundamental frequency and a_n and b_n are the sine and cosine amplitudes of the n -th harmonic of frequency nf .

Such a decomposition is called a *Fourier series*. From the Fourier series the function can be reconstructed; i.e., if the period T , is known and the amplitudes are given, the original function of time can be found by performing the sums of equation 2.3.

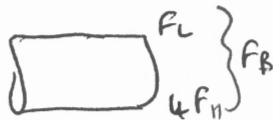
A *data* signal, such as a byte consisting of eight bits that has a finite duration, which all of them do, can be handled by just imagining that it repeats the entire pattern forever, i.e. the behaviour during the interval from T to $2T$ is the same as from 0 to T , etc.

The a_n amplitudes can be computed for any given $g(t)$ by multiplying both sides of Eqn. 2.3 by $\sin(2\pi k f t)$ and then integrating from 0 to T . Since

$$\int_0^T \sin(2\pi k f t) \sin(2\pi n f t) dt = \begin{cases} 0, & \text{if } k \neq n \\ T/2, & \text{if } k = n. \end{cases}$$

only one term of the summation survives: a_n . The b_n summation vanishes completely. Similarly, by multiplying equation 2.3 by $\cos(2\pi k f t)$ and integrating between 0 and T , we can derive b_n . By just integrating both sides of the equation as it stands, c can be found. The results of performing these operations are as follows:

$$\begin{aligned} a_n &= \frac{2}{T} \int_0^T g(t) \sin(2\pi n f t) dt \\ b_n &= \frac{2}{T} \int_0^T g(t) \cos(2\pi n f t) dt \\ c &= \frac{2}{T} \int_0^T g(t) dt \end{aligned}$$



Example 2.1 Consider a specific example, the transmission of the ASCII character "b" encoded in an 8-bit byte. The bit pattern to be transmitted is 01100010. The Fourier analysis of this signal yields the coefficients:

$$\begin{aligned} a_n &= \frac{1}{\pi n} [\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)] \\ b_n &= \frac{1}{\pi n} [\sin(3\pi n/4) - \sin(\pi n/4) + \sin(7\pi n/4) - \sin(6\pi n/4)] \\ c &= 3/4 \end{aligned}$$

if we increase the capacity (bits per second) the frequency of the fundamental increases.

2.3 Transmission Media

bandwidth = quality of road capacity = speed limit imposed on road

The transmission of an electrical signal obviously has to be along an electrical conductor or medium. The type of transmission medium used is important, since it determines the maximum rate, in terms of binary digits (bits) per second or Bps, that data can be transmitted. In all the instances that follow, the signal is in fact an analog signal modulated (also said to be "coded") with digital data which we shall learn about in Sec. 2.5. Pure digital signals cannot travel very far along a conductor before losing their shape as we shall see in Sec. 2.4 below. The maximum distance such signals can travel is about 50 meters and frequently no further than from your computer to the printer.

Some of the more common types of transmission media are mentioned next.

Holding bandwidth of conductor F_B constant, and increasing the capacity, the number of harmonics (i.e. $nf \leq F_B$) will decrease and the quality/shape of the digital signal will deteriorate.

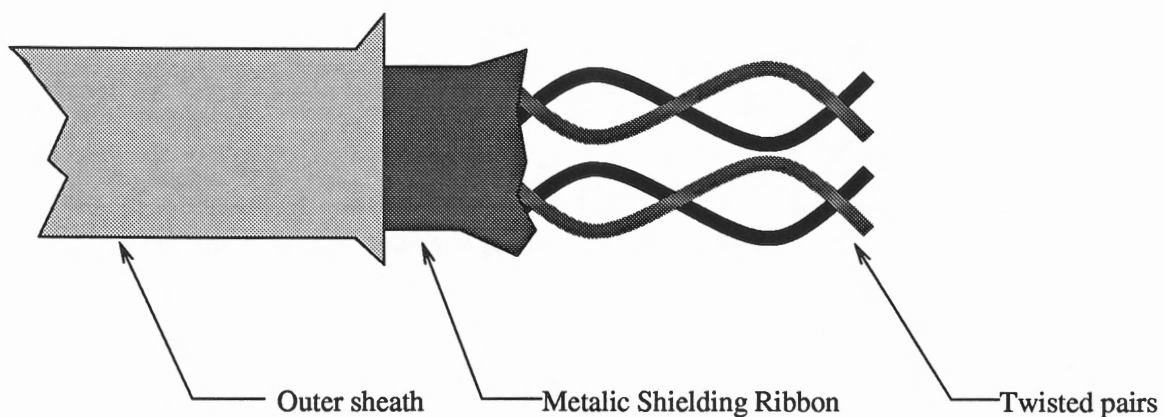


Figure 2.3: Shielded twisted-pair line

2.3.1 Twisted pair

A two-wire open line, is the simplest type of transmission medium illustrated in Fig. 2.3. In such a line, each wire is insulated from the other and both may be open to free space, when it is called an “Unshielded Twisted Pair(UTP)”.

With this type of line, care is needed to avoid cross coupling of electrical signals between adjacent wires in the same cable. This is known as *crosstalk* and is caused by capacitive coupling between the two wires. In addition the open structure of this type of line makes it susceptible to the pick-up of spurious noise signals from other electrical signal sources caused by electromagnetic radiation. The main problem with interference signals of this type is that they may be picked up in just one wire - the signal wire, for example - and not the ground wire. As a result, an additional difference signal can be created between the two wires and, since the receiver normally operates using the difference signal between the two wires, this can give rise to an erroneous interpretation of the combined (signal plus noise) received signal.

Twisting the wires helps to avoid the interference mentioned above. The resulting close proximity of both the signal and ground reference wires means that any interference caused by extraneous signal sources is picked up by both wires and hence its effect on the difference signal is reduced. Furthermore, if multiple twisted pairs are enclosed within the same cable, the twisting of each pair within the cable further reduces interference effects caused by crosstalk.

Another way of limiting the interference is to cover the twisted pair with a conducting material such as aluminum foil to *shield* the lines, whence it is not surprisingly called a “*Shielded Twisted-Pair*” or *STP* cable.

The main limiting factor of a twisted pair line is caused by a phenomenon known as the *skin effect*: as the bit rate (and hence frequency) of the transmitted signal increases, the current flowing in the wires tends to flow only on the outside surface of the wire, thus using less of the available cross-section. This has the effect of increasing the electrical resistance of the wires for higher frequency signals which in turn causes more attenuation of the transmitted signal. In addition, at higher frequencies, an increasing amount of signal power is lost due to radiation effects.

There are various categories of UTP cable prescribed by industry Standard EIA/TIA-568 ranging from Category 1 through 5E. Categories 3 and 5 are mostly used for data transmission and with speeds ranging to 100 Mbps Category 5E “E” for “Enhanced” is more commonplace. The reader is referred to any standard text such as that by Gallo and Hancock ([GalHan] page 117) for a closer description of the properties of the various categories.

4 levels = 2 bits

each signal represents a baud rate of 8,
and a bit rate of 16 bps

baud - change in signal
bps - bits represented by a level

One type of transmission line that minimizes both crosstalk and attenuation effects is the coaxial cable.

2.3.2 Coaxial cable

The most popular Local Area Network (see Chapter 8) transmission media are *baseband* (digital) and *broadband* (analog) coaxial cable. The cable is physically similar in both implementations, but the electrical characteristics are generally different.

In a coaxial cable, the signal and ground reference wires take the form of a solid centre conductor running concentrically (coaxially) inside a solid (or braided) outer circular conductor as shown in Fig. 2.4. The space between the two conductors should ideally be filled with air, but in practice it is normally filled with a dielectric insulating material with either a solid or honeycomb structure.

Due to its geometry, the centre conductor is effectively shielded from external interference signals and also only minimal losses occur due to electromagnetic radiation and the skin effect. Coaxial cable can be used with a number of different signal types, but typically 10 or even 20 Mbps over several hundred metres is perfectly feasible. Also, coaxial cable is applicable to both point-to-point and multi-point topologies.

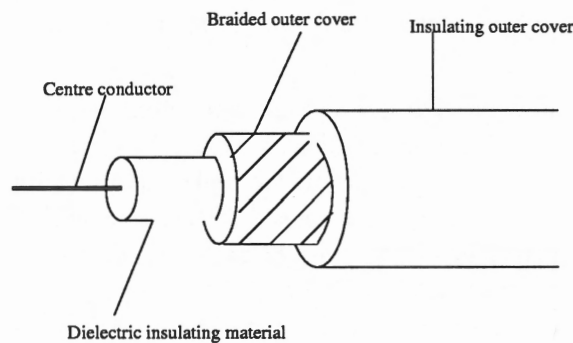


Figure 2.4: A typical coaxial cable.

microwave: 4.2 GHz

2.4 Channel Characteristics

Electrical signals traveling along a conductor are subjected to various attenuation and distortion effects as illustrated diagrammatically in Fig. 2.5. As can be seen, a signal transmitted across any form of transmission medium will be attenuated and affected by limited bandwidth, delay distortion and noise. Each is always present and will produce a combined effect. We consider each of these briefly.

2.4.1 Attenuation

As a signal propagates along a transmission link, line or channel² its amplitude decreases. This is known as *signal attenuation*. Normally, to allow for attenuation, a definite limit is set on the length of the line to ensure that the electronic circuitry at the receiver will reliably detect and interpret the received attenuated signal. If the length of line exceeds this limit, one or more *amplifiers* or *repeaters* must be inserted at the set limits along the line to restore the received signal to its original level.

²These terms are all synonymous.

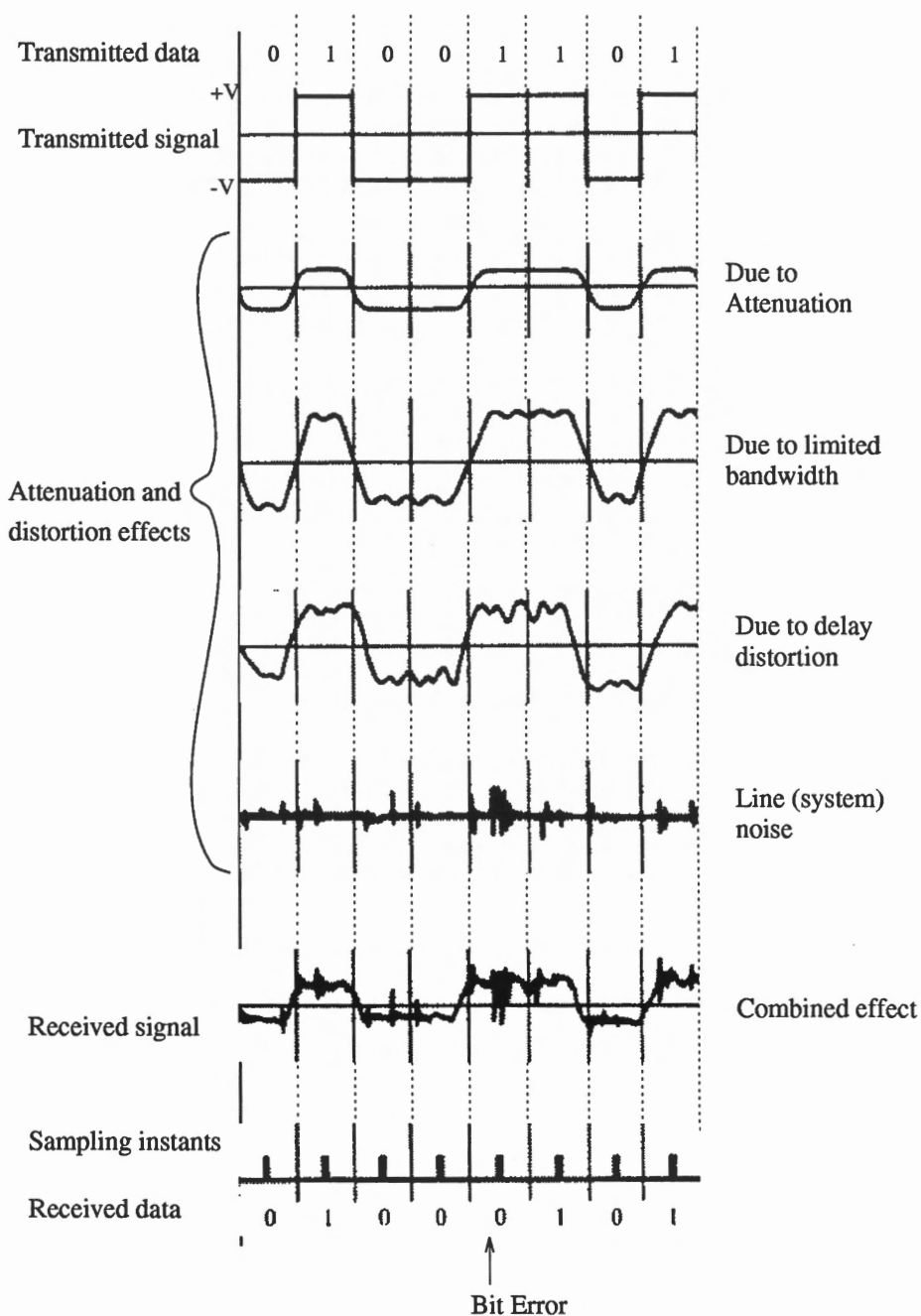


Figure 2.5: Sources of attenuation and distortion of a digital signal

The amount of attenuation a signal experiences also increases with frequency. Hence, since a transmitted signal comprises a range of different frequency components, as explained in Sec. 2.2, this has the additional effect of distorting the received signal. This problem is overcome by devices, known as *equalizers*, which boost different frequencies differently to equalize the amount of attenuation across a defined band of frequencies.

2.4.2 Delay distortion

The rate of propagation of a sinusoidal signal along a transmission line with the frequency of the signal. Consequently, when transmitting a digital signal, the various frequency components making up the signal arrive at the receiver with varying delays between them. The effect of this is that the received signal is distorted. The amount of this *delay distortion* increases as the bit rate of the transmitted data increases, so some of the frequency components associated with each bit transition are delayed and start to interfere with the frequency components associated with a later bit. Delay distortion is therefore also known as *intersymbol interference* and its effect is to vary the bit transition instants of the received signal as illustrated in Fig. 2.5. Consequently, since the received signal is normally sampled at the nominal centre of each bit duration, the delay distortion may lead to the incorrect interpretation of the received signal as the bit rate increases.

2.4.3 Electrical noise

Much as for humans, electrical noise is any undesirable signal present on a transmission medium which is carrying useful information. There are two forms of electrical noise: Thermal noise is always present and is an inherent property of receivers and transmitters. Thermal noise is a function of temperature and cannot be eliminated. The amount of thermal noise in a bandwidth of 1 cps in a conductor is

$$N_o = kT$$

Watt/cycle where

$$N_o = \text{noise power density in watts per cycle of bandwidth}$$

$$k = \text{Boltzmann's constant}$$

$$= 1.3803 \times 10^{-23} \text{ joules/}^\circ \text{K}$$

$$T = \text{temperature in degrees Kelvin}$$

Thermal noise is also known as *line noise* illustrated conceptually in Fig. 2.5. In the limit, as a transmitted signal becomes attenuated, the signal will reach the same level as the line (background) noise level.

An important parameter associated with a transmission medium, therefore, is the ratio of the power in a received signal, S , to the power of the noise level, N . The ratio S/N is known as the *signal-to-noise* ratio and normally is expressed in *decibels* or *dB* as

$$\frac{S}{N} = 10 \log_{10} \left(\frac{S}{N} \right) \text{ dB}$$

A signal to noise ratio of 10 is 10 dB, a ratio of 100 is 20 dB, a ratio of 1000 is 30 dB and so on.

A high S/N ratio means a high signal value relative to the prevailing noise level, resulting in a good quality signal. Conversely, a low S/N ratio means a low quality signal. The theoretical maximum information rate of a transmission medium is related to the S/N ratio by a formula first stated in 1940 by Shannon.

Shannon's law for noisy channels states that the maximum data rate C of any channel whose bandwidth is B Hz, and whose signal-to-noise ratio is S/N , is given by

$$C = B \log_2(1 + S/N).$$

Example 2.2 A channel of 3000-Hz bandwidth ($B = 3000$), and a signal to thermal noise ratio of 1000 (typical parameters of the telephone system) can never transmit more than

$$\begin{aligned} C &= B \log_2(1 + S/N) \\ &= 3000 \log_2(1001) \\ &\doteq 3000 \times 10 \end{aligned}$$

or 30 000 Bps.

Shannon's law was derived using information-theory arguments and has very general validity. It should be noted, however, that this is only an upper bound. In practice, it is difficult to even approach the Shannon limit. A bit rate of 9600 bps on a voice-grade line is considered excellent, and is achieved by sending 4-bit groups at 2400 baud.

In practice there is another source of noise known as *impulse noise* caused by external influences such as fluorescent light transformers, electrical equipment such as elevators and door switches. Impulse noise is generally only a minor annoyance for analog data such as voice transmission where it would be manifested by a few clicks or crackles and no loss of information. It is different for digital data however where a sharp spike of noise energy lasting say 10 milliseconds would have no effect on voice data, but would be the death knell of 640 bits being transmitted at 64 KBps. Although impulse noise is always a threat, much of it can be eliminated through proper shielding and cable installation.

2.4.4 Bandwidth limitation

No transmission line can transmit signals without the signals losing some power in the process. If all the harmonic frequencies (see Sec. 2.2) were attenuated equally, unlike explained in the previous section, the resulting signal would be reduced in amplitude, but not distorted, i.e., would have the same nice squared shape as that shown at the top of Fig. 2.6. Usually, harmonics of all frequencies less than some cutoff frequency f_B say, are transmitted almost unattenuated, with all harmonics with frequencies higher than f_B strongly attenuated. The frequency f_B is called the *bandwidth* of the link.

For the explanations to follow we consider the time it takes for the binary signal to repeat itself to be the period $\frac{1}{f_B}$ of the fundamental or first harmonic. In the example of Fig. 2.6 the period of the fundamental harmonic is considered to be the duration of two bits of the binary signal. In some cases, the bandwidth is a physical property of the transmission medium, and in other cases special electrical circuitry, called a *filter*, is introduced into the circuit to limit the amount of (scarce) bandwidth available to each signal.

Consider the arbitrary square wave signal in Fig. 2.6. If the bandwidth were so narrow that it would pass only the fundamental frequency, we would get the simple sinusoidal wave shown at the top. If however it would let 3 harmonics through, i.e., frequencies up to three times that of the fundamental, we would get the second signal being added with a different amplitude to the original signal. With 5 harmonics we would have the signal third from the top, with 7 harmonics the second last signal in the figure. Despite it looking quite noisy, it now bears more than a passing resemblance to a square wave. In fact, if we add more and more harmonics, we would get closer and closer to a square wave until, if the bandwidth were infinite (which it never is) we would have the square wave back.

The terms "bandwidth" (measured in cps or Hertz) and capacity (measured in Bps) are frequently used interchangeably in industry. They mean entirely different things as you now know, but to understand that they express different properties of a link, consider the binary signals in Fig. 2.7. In general,

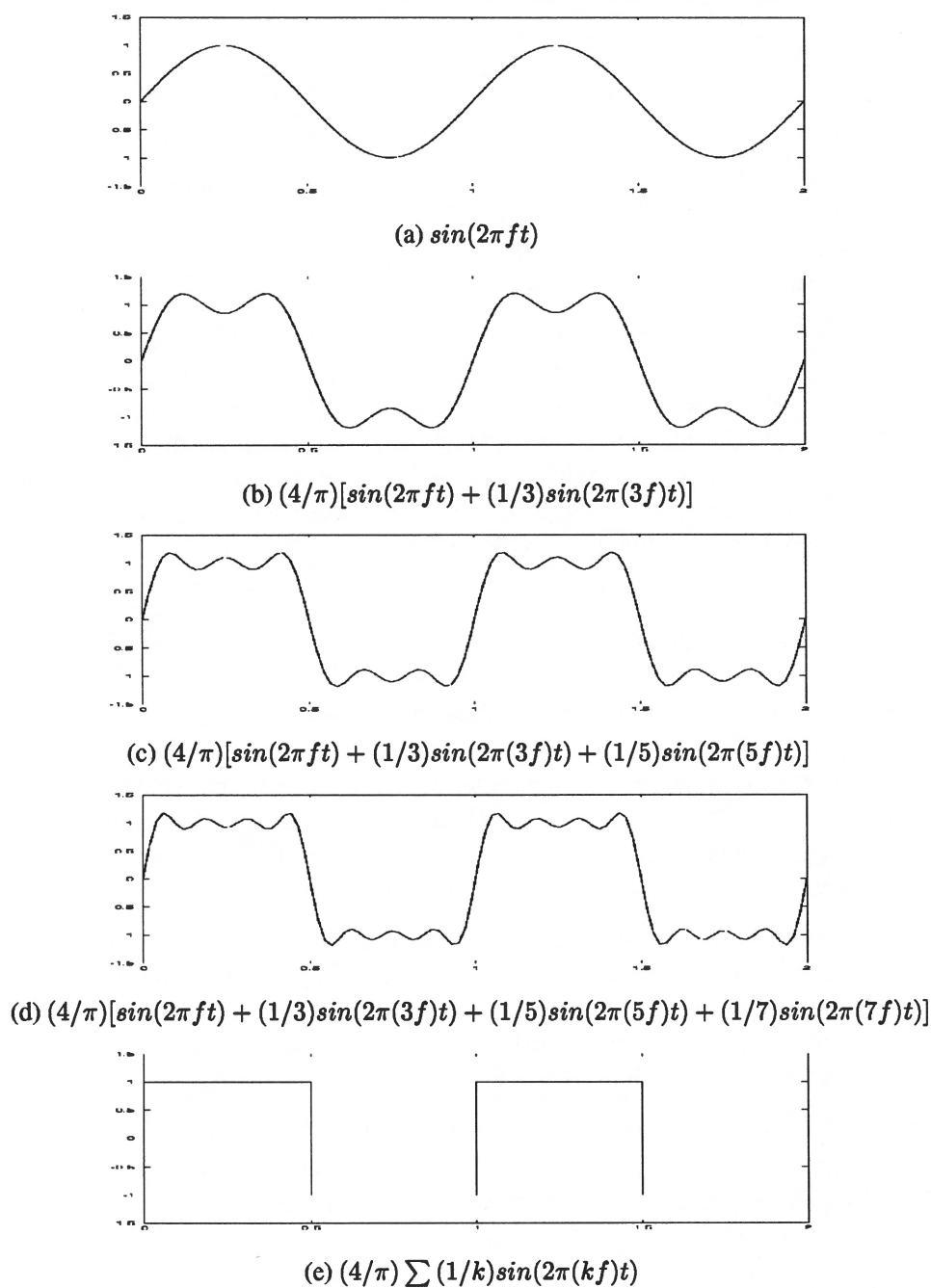


Figure 2.6: A binary signal and its Fourier components.

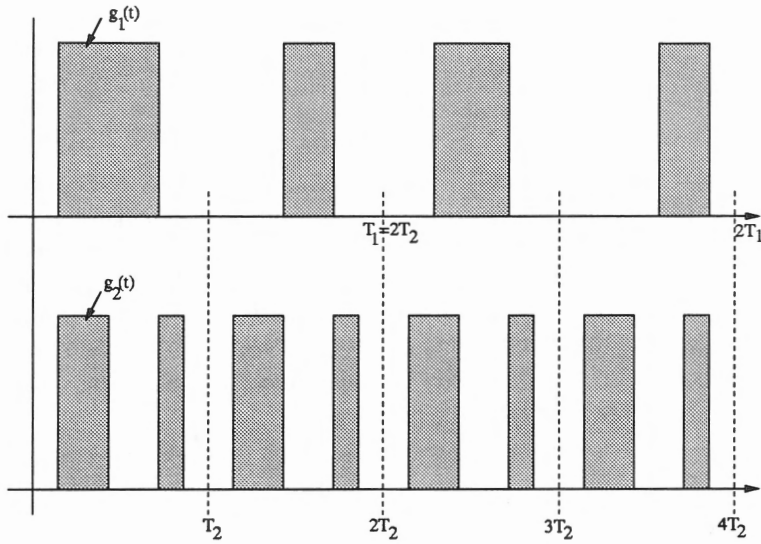


Figure 2.7: Two binary signals: $g_2(t)$ with double the *capacity* (number of bits per second) of $g_1(t)$.

$$g(t) = \frac{c}{2} + \sum_{n=1}^{\infty} a_n \sin\left(\frac{2\pi n}{T}t\right) + \sum_{n=1}^{\infty} b_n \cos\left(\frac{2\pi n}{T}t\right)$$

Hence for $g_2(t)$ we write

$$\begin{aligned} g_2(t) &= \frac{c_2}{2} + \sum_{n_2=1}^{\infty} a_{n_2} \sin\left(\frac{2\pi n_2}{T_2}t\right) + \sum_{n_2=1}^{\infty} b_{n_2} \cos\left(\frac{2\pi n_2}{T_2}t\right) \\ &= \frac{c_2}{2} + \sum_{n_2=1}^{\infty} a_{n_2} \sin(2\pi n_2 f_2 t) + \sum_{n_2=1}^{\infty} b_{n_2} \cos(2\pi n_2 f_2 t) \end{aligned}$$

and for signal $g_1(t)$,

$$\begin{aligned} g_1(t) &= \frac{c_1}{2} + \sum_{n=1}^{\infty} a_n \sin\left(\frac{2\pi n}{T_1}t\right) + \sum_{n=1}^{\infty} b_n \cos\left(\frac{2\pi n}{T_1}t\right) \\ &= \frac{c_1}{2} + \sum_{n=1}^{\infty} a_n \sin\left(\frac{2\pi n}{2T_2}t\right) + \sum_{n=1}^{\infty} b_n \cos\left(\frac{2\pi n}{2T_2}t\right) \end{aligned}$$

Hence, if we have a channel with fixed bandwidth say f_B , the number m_1 of harmonics of frequency $m_1 f_1$ of signal $g_1(t)$ with *fundamental* frequency f_1 which will be allowed through are:

$$\begin{aligned} m_1 f_1 &\leq f_B \\ \text{or } m_1 &\leq \frac{f_B}{f_1} \\ &= f_B T_1 \end{aligned}$$

and the number of harmonics m_2 of frequency $m_2 f_2$ of signal $g_2(t)$ of fundamental frequency f_2 which will be allowed through are:

$$\begin{aligned} m_2 f_2 &\leq f_B \\ \text{or } m_2 &\leq \frac{f_B}{f_2} \end{aligned}$$

$$\begin{aligned}
&= f_B T_2 \\
&= \frac{f_B T_1}{2} \\
&= \frac{m_1}{2}
\end{aligned}$$

Note that the bandwidth f_B need not be an integer multiple of the fundamental frequency f_1 .

In other words, only half as many harmonics will be allowed through in the case of signal $g_2(t)$ with twice the capacity (bps) of $g_1(t)$. Thus, for *fixed bandwidth* (f_B) the more bits per second, the worse the signal quality. Alternatively, to maintain the signal quality, the higher the capacity, the higher the bandwidth has to be. The terms “bandwidth” and “capacity” are thus used synonymously but mean quite different things electrically. Figure 2.8 further illustrates the principle.

2.5 Data encoding

Now that we understand analog signals and their properties and digital signals and Fourier analysis it is possible to learn about the way data can be encoded. One way that we saw in Sec. 1.4 is to simply send the data as a binary signal by using Manchester (en-)coding. Manchester encoding is just one of many encoding techniques as we will now briefly discuss.

2.5.1 Digital Data, Digital Signals

The simplest way to transmit digital signals is obviously to use two different voltage levels for the two binary digits. Codes that work this way share the property that the voltage level is constant during a bit interval – i.e., there is “no return to a zero voltage level” (during a bit interval). Most commonly a negative voltage is used to represent one binary value and a positive voltage level to represent the other. This code is known as *Nonreturn to Zero-Level* (NRZ-L) and is illustrated in Fig. 2.9. An interesting variation of NRZ is known as *Nonreturn to Zero, Invert on ones* (NRZI). NRZI is similar to NRZ-L in that the signal maintains a constant voltage for the duration of a bit interval. However, now the *data are encoded as the presence or absence of a signal transmission at the beginning of the bit time*. A transition from low to high or high to low at the start of a bit interval indicates a binary one for that bit interval;

the absence of a transition a zero bit value.

NRZI is an example of *differential encoding*. In differential encoding the data to be transmitted are represented as a change between two successive data symbols rather than the signal levels themselves. In general: If the bit to be encoded next is a binary 0, then the bit is encoded with the same signal (level) as the preceding bit; if the bit to be encoded is a binary 1, then the bit is encoded with a different signal than the preceding bit.

One benefit of differential encoding is that it is easier to detect a transition in the presence of noise than to keep track of the signal level. Another benefit for the technician who has to install hundreds of twisted-pair lines for instance, is that if the leads from the pair are inadvertently inverted all 1s and 0s for NRZ-L will be inverted. This is not possible with differential encoding.

Nevertheless, because of the NRZ encoding techniques there will always be a non-zero direct current (dc) component in the signal and synchronization of the receiver and transmitter remains a problem should the timing (when do each read what bit interval) between the receiver and transmitter shift.

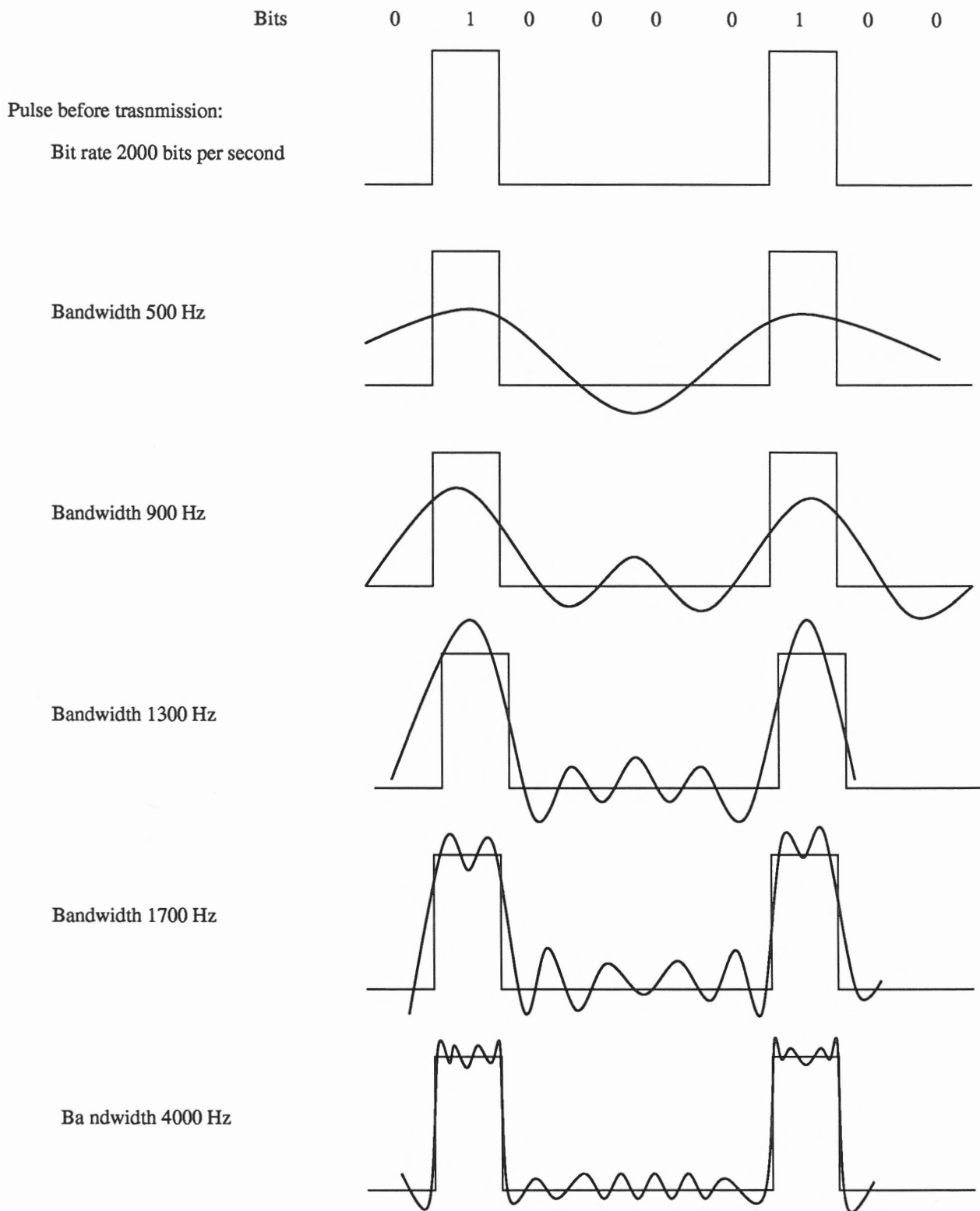


Figure 2.8: Illustrating the effect of bandwidth on a digital signal

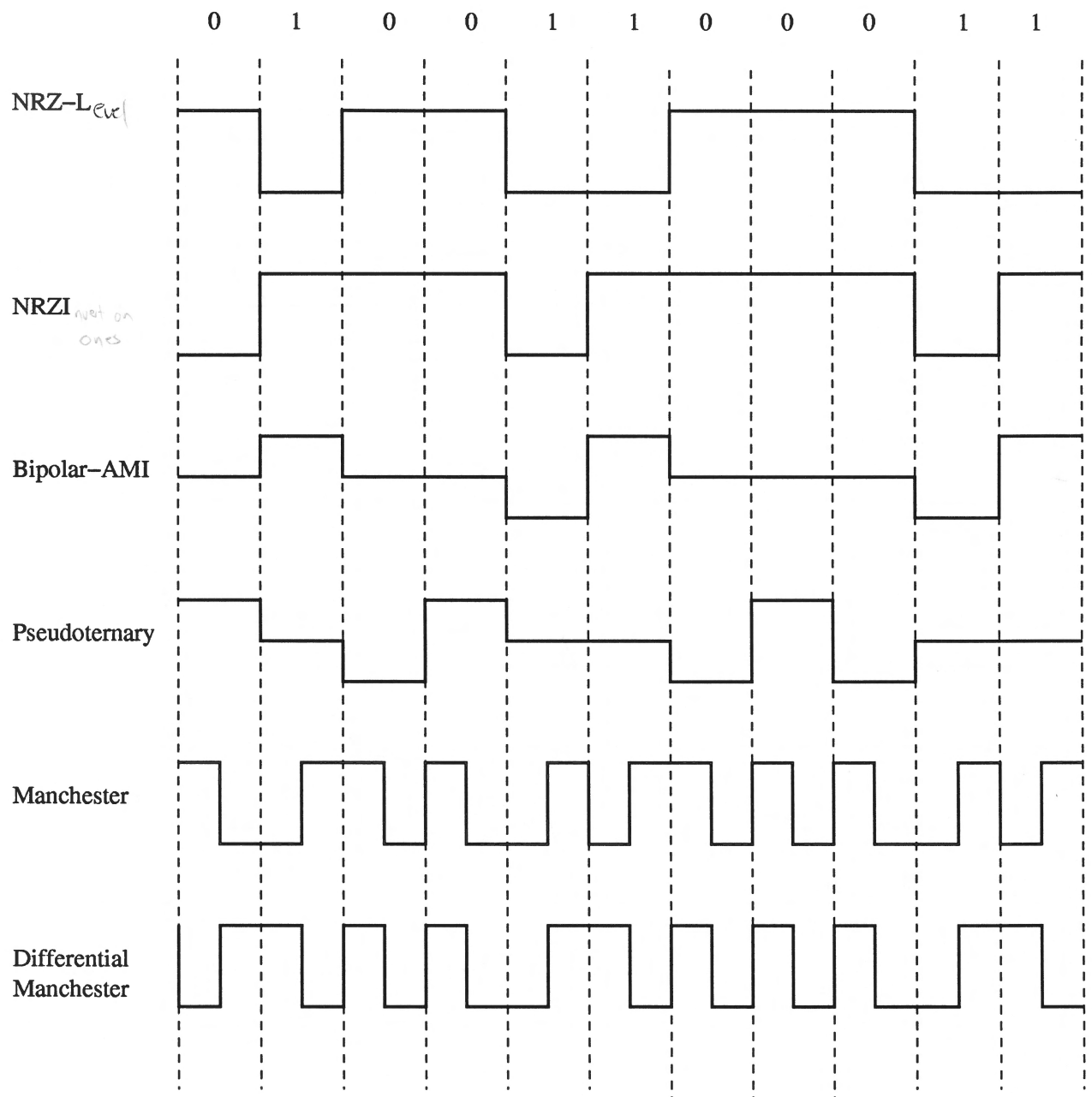


Figure 2.9: Some digital encoding techniques

Dif-man if there is no transition at the beginning of the interval, it is 1
 if there is a change, 0
 we always change in the middle. for synchronization
 benefit - we can reverse polarity, signal still comes through.

Another family of coding techniques that eliminate these NRZ problems is known as *Biphase* techniques. One of these, *Manchester encoding* we have seen already and it is illustrated again in Fig. 2.9. In the Manchester code, there is a transition at the middle of each bit interval. The transition at the mid-interval instant serves to synchronize the sender and receiver and also represents the data: a low-to-high transition represents a binary 1, a high-to-low represents a binary 0. In *Differential Manchester encoding* the mid-interval transition is for synchronization only. The encoding of a 0 is represented by the presence of a transition at the beginning of a bit interval, a 1 by the absence of a transition.

Reflecting on what we learned in Sec. 2.2 we realise immediately that both Manchester encoding methods require twice the bandwidth to send at the same data rate (Think!). Yet they have the advantage that

- there is always a voltage transition in each bit interval that the receiver can synchronize upon. The biphase codes are known as self-clocking codes for this very reason.
- There is no direct current component.
- The absence of a known transition would signal an error. A noisy line would have to invert the signal both before and after the transition to cause an error to go undetected.

2.5.2 Digital Data, Analog Signals

From sec. 2.4 we know already that digital signals cannot travel long distances along a conductor before they get distorted to the point that they are useless. And obviously digital signals cannot travel through the air as electro magnetic waves can. In order to make digital transmission over long distances possible we code, or *modulate* the analog signal with the digital data as shown in Fig 2.10. The function of a *modulator* is to provide a one-to-one mapping of the i data symbols into a set of signals $S_i(t)$, $i = 1, 2, \dots, N$. In general, the signal $S_i(t)$ may be represented by

$$\begin{aligned} S_i(t) &= A_i(t) \sin(\omega_i t + \theta_i(t) + \theta_0) \\ &= B_i(t) \cos(\omega_i t + \theta_0) + C_i \sin(\omega_i t + \theta_0) \\ \text{where } A_i(t) &= \sqrt{B_i^2(t) + C_i^2(t)} \\ &= \theta_i(t) = \tan^{-1} \left[\frac{B_i(t)}{C_i(t)} \right] \end{aligned}$$

The waveforms $A_i(t)$, $B_i(t)$, $C_i(t)$ and $\theta_i(t)$ are all known functions of time, $f_0 = \omega_0/2\pi$, is the *carrier frequency* and θ_0 is the *carrier phase*.

If either the amplitude, $A_i(t)$, the frequency, f_0 , or the phase, $\theta_i(t)$, of $S_i(t)$ is varied discretely with known time functions, one has the various forms of digital modulation.

When the amplitudes $A_i(t)$, $i = 1, 2, \dots, N$, are pulse waveforms while ω_0 and θ_0 are held constant, this discrete form of amplitude modulation is commonly referred to as *pulse-amplitude modulation (PAM)*.

Other pulse modulation techniques include *pulse-position modulation (PPM)* and *pulse-duration modulation (PDM)*. In the former, data are conveyed by varying the position in time of a pulse waveform, while in the latter, data are conveyed by varying the duration of the pulse.

If the frequency of signal $S_i(t)$ is varied discretely with both $A_i(t)$ and $\theta_i(t)$ held constant – that is, $\omega_i = 2\pi(k + i)/T$ for $i = 1, 2, \dots, N$, k any integer, the modulation technique is called *multiple frequency-shift-keying (MFSK)*.

Need to convert digital signals to analogue cos - can't transmit binary through cable > Som need to use electromagnetic spectrum (not digital)

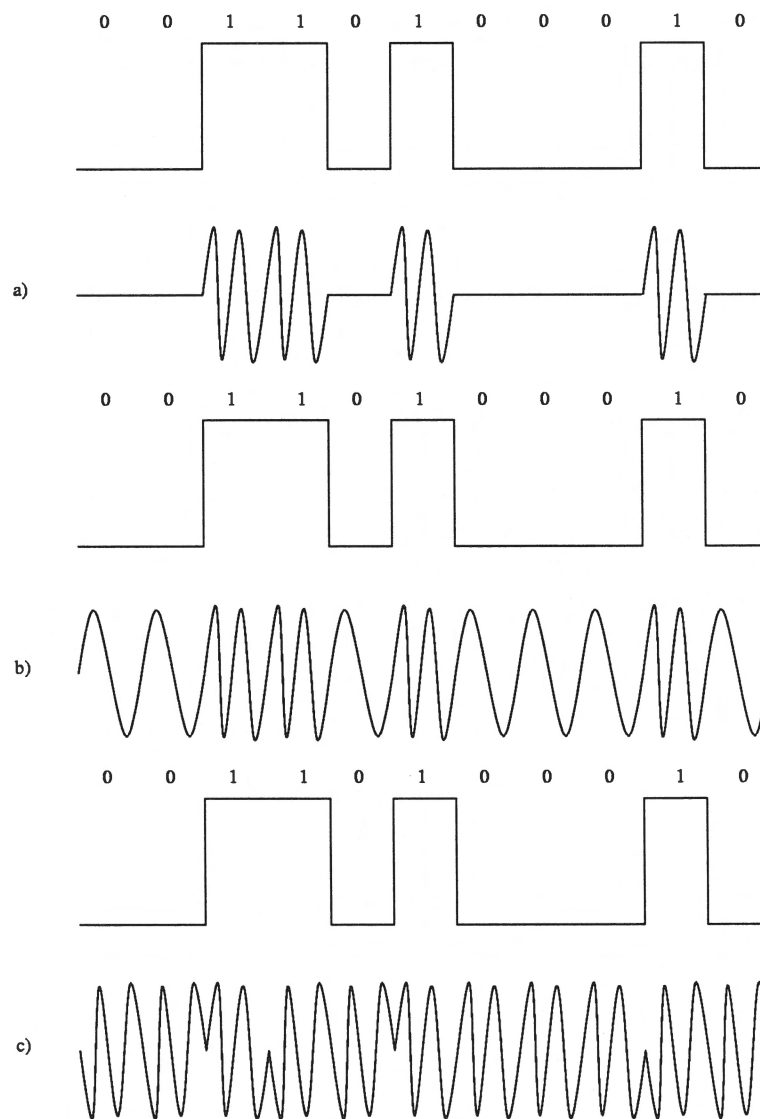


Figure 2.10: A binary signal (a) amplitude modulated (PAM) (b) frequency modulated (MFSK) (c) phase modulated (PSK)

If the phase $\theta_i(t) = 2(i - 1)\pi/N$ for all $i = 1, 2, \dots, N$ and both $A_i(t)$ and ω_i are held constant, we have *multiple phase-shift-keying (MPSK)*. If $N = 2$, MFSK and MPSK are usually referred to as *frequency-shift-keying (FSK)* and *phase-shift-keying (PSK)* respectively. Finally, if the amplitude and phase of a constant frequency carrier are varied discretely, one has the hybrid of PAM and MPSK modulation called combined *amplitude-phase-shift-keying (APSK)*.

FSK is the most common form of modulation of transmission rates of up to 1800 bps and is less susceptible to noise than PAM. PSK is again less susceptible to noise than PAM and is used in equipment for transmission at speeds above 2000 bps. In the most common form of PSK, the carrier wave is systematically shifted 45, 135, 225, or 315 degrees at uniformly spaced intervals. Each phase shift transmits 2 bits of information. Details of all these techniques are not necessary to understand what networks are about.

At the receiver an operation which is the inverse of modulation is performed. In any two-way communication the a modulator must also demodulate so that combination device is called a *modem* in either case.

2.5.3 Analog Data, Digital Signals

Historically, telephone networks carried only analog voice signals. When digital data came along, ways had to be found to carry the digital data over the analog (signal) networks using a modem, as we now know. With the increase need for data networks the obvious question is why one cannot carry analog signals over digital networks and thus avoid the need for two different networks altogether? This is increasingly being done to the point that no analog networks are being built anymore and that at some distant time in the future all communication will be digital.

The technique to transmit analog signals over a digital channel (still using modulation, note) used is called *Pulse Code Modulation (PCM)* or *sampling*. Sampling requires that the amplitude of the analog signal be measured at regularly spaced intervals as illustrated in Figure 2.11.

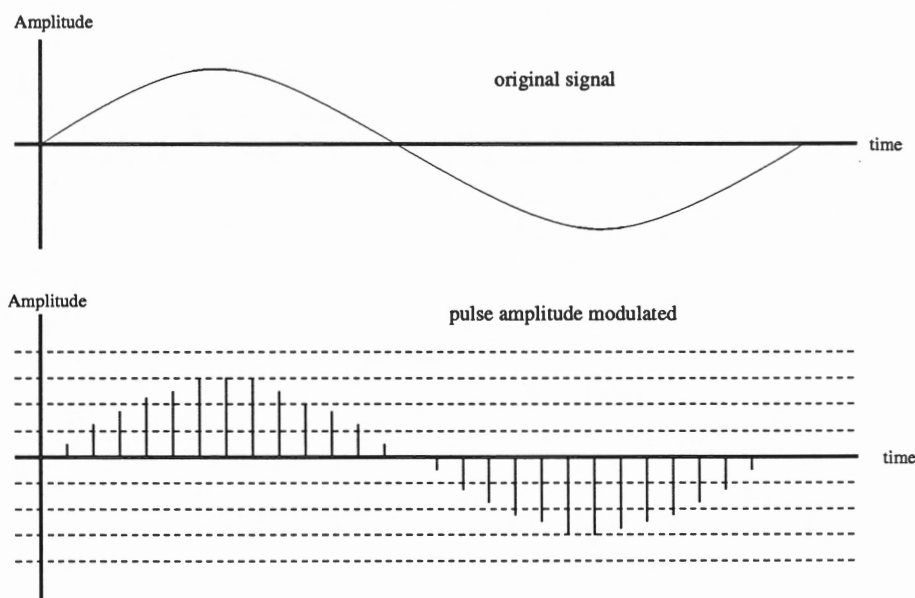


Figure 2.11: Illustrating the principles of Pulse Code Modulation (PCM)

At each sampling point the measured amplitude is coded as a binary number and the digital data are sent over our digital network to the destination where the receiver re-constructs the analog signal from the digital data.

Nyquist an analogue signal of frequency F can be completely reconstructed by sampling at a rate of $2F$.

Voila, analog signals over a digital network. More specifically, each of the levels used in the digitisation, is known as a quantisation level and is coded by 8 bits. This gives us 128 negative quantisation and 128 positive levels. The levels are not linearly spaced but rather logarithmically. This means the spacing at the larger amplitudes are much wider than at the smaller amplitudes. The reason for this is that a linear scale (such as the one shown) is unable to accurately account for low amplitude signals or small changes in signal. The question next arises as to how often should one sample the analog signal?

The answer to this was provided by an engineer called Nyquist from the early half of the 20th century. Nyquist's theorem states that an analog signal can be reconstructed accurately by sampling it at a rate equal to exactly twice the highest frequency of the analog signal (perhaps you now have an idea what the 4-bit over-sampling means on your CD player). If the signal consists of M discrete levels, Nyquist's law states that the maximum data transfer rate C , measured in bits per second, of a noiseless line is given by:

$$C = 2B \log_2 M$$

\hookrightarrow Bandwidth of channel

The next obvious question is what the highest frequency is that we need to consider. If we are concerned with voice data, the frequency that the (young) human ear can detect ranges from about 30 to 20 000 cps. However, the human voice is not much distorted by limiting the range of frequencies (the bandwidth) between 300Hz and 3400Hz a bandwidth of 3100 cps (have you ever wondered why the human voice sounds different over the telephone than through open space?). For technical reasons (to avoid interference from adjacent channels) the bandwidth is chosen to be 4000 cps and so the sampling rate will have to be twice this frequency. This means that in a telephone network our voice is sampled exactly 8000 times a second and generating 8 bits of data per sample. This gives a signal rate 64000 bits per second (or 64Kbps). With digital bandwidth becoming increasingly inexpensive and plentiful there is no longer a need to limit the voice bandwidth. However, all telephone equipment use this standard and with the communication equipment from our homes to the local exchange (called the local loop) likely to remain analog for many years, the 64Kbps is part of our digital world forever.

Unfortunately, like so much else there are two standards for PCM encoding. There is the European standard, which we have just described, and the US/Japan standard which instead of uses 7 bits instead of 8. A signal coded this way of course have a rate of 7 multiplied by 8000 which gives 56000 (56Kbps) bits per second! Both systems (64K signals or 56K signals) represent one channel. A channel represents the bit stream generated by sampling one conversation. With digital technologies, it is possible to assemble these many of these channels into a faster single bit stream. This process is known as multiplexing. The European standard assembles 32 channels into a single stream at what is called the primary level and generates a 2.048 Mbps signal (also known as an E1 line). The US/Japan assembles 24 into a 1.554 Mbps signal (where it is known as an T1 line). Signals using the two standards need to have an adaptor to convert between the two standards. We need not stop at this point and we can subsequently continue to multiplex a primary rate signal into a higher order signal. The European version is 8 Mbps (equals 128 channels) with the US/Japan version being approximately 6 Mbps and so on. It is not uncommon at all to have the backbone communication networks with a single connection carrying well over 60000 channels or 2.5Gbps. Such is the power (and future) of digital communications.

Finally it is clearly possible to modulate an analog (carrier) signal with analog data such as audio signals. Most of us who have listened to a radio know the difference between AM and FM.

2.6 Exercises

Exercise 2.1 If the human voice can be safely restricted to the range 0-4000 cycles, why is the capacity for a voice line 64Kbps? (Hint: assume a 256 level PCM.) [4]

Exercise 2.2 Name three ways a signal can be modulated for transmission.

[3]

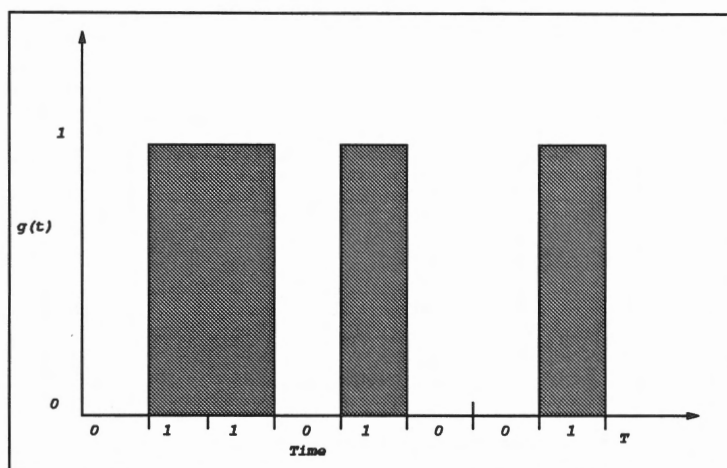


Figure 2.12: A transmitted byte

Exercise 2.3 In the figure above, what are the Fourier coefficients a_n , b_n and c_n ?

Exercise 2.4 What is the cosine amplitude of the 5th harmonic in the above example?

Exercise 2.5 A network uses copper cabling as its transmission medium. State two ways of 'tapping' the network lines. How can you ensure that security is not compromised even if the lines are 'tapped'?

Exercise 2.6 If the human voice can be safely restricted to the range 0–4000 cycles, why is the capacity for a voice line 64Kbps? (Hint: assume a 256 level PCM.)

Exercise 2.7 A controlling system in a brewery receives temperature readings from a number of vats. Assuming that the temperature ranges from 0–100 degrees, and the readings are accurate to the nearest $1/2$ a degree, how many bits are needed to adequately represent the readings? Assume also that there are five different vats. What is the minimum capacity required if a reading needs to be taken from all the vats every millisecond? What is the baud rate of the system? What is the bandwidth of the system if 8 harmonics are used to represent the binary signals?

Chapter 3

Optic Signals

3.1 Objectives of this Chapter

This chapter is about optic signals and optic fibres which are fast becoming the most dominant transmission technology. The reasons should be clear on understanding the material to follow.

1. What are optic signals and how do we generate them? This is the starting point of this chapter.
2. As in the case of electrical signals, optic signals have a medium of their own. The technology of optic fibres has made huge strides in the past decade and still does.
3. Life is never perfect however, and optic signals suffer from degradation as do electrical signals, but for different reasons obviously. We explain signal loss in optic fibre in the last section of this chapter.

There is a huge amount to know about optic signals which we do not cover and a good source of tutorial material is www.corning.com. What we do cover is sufficient though for you, the reader, to understand the various optic standards and technology developments.

Optic signals can never carry more than
one bit of info at any given instant \rightarrow
band rate of 1.

Optic signals, albeit of a very different frequency, were first used around 420 BC in Ancient Greece to send coded signals. Charles Babbage, the man who designed the analytical engine, also invented the “light-flashing machine” which in his words worked as follows:

I then, by means of a small piece of clock-work and an argand lamp, made a numerical system of occultation, by which any number might be transmitted to all those within sight of the source of light.

Probably one of the first communication networks ever was the *optic telegraph* illustrated in Fig. 3.1 designed by Claude Chappe in France in 1791. This pioneering French telegraph was followed by a Swedish design and soon a large number of different designs in other countries. A network of such optic telegraphs was built in 1795–96 connecting Stockholm and Eskerö on Åland in Sweden with stations at most 14 km apart. The transmitter consists of ten flaps, 9 of them arranged in a three-by-three matrix and a tenth on

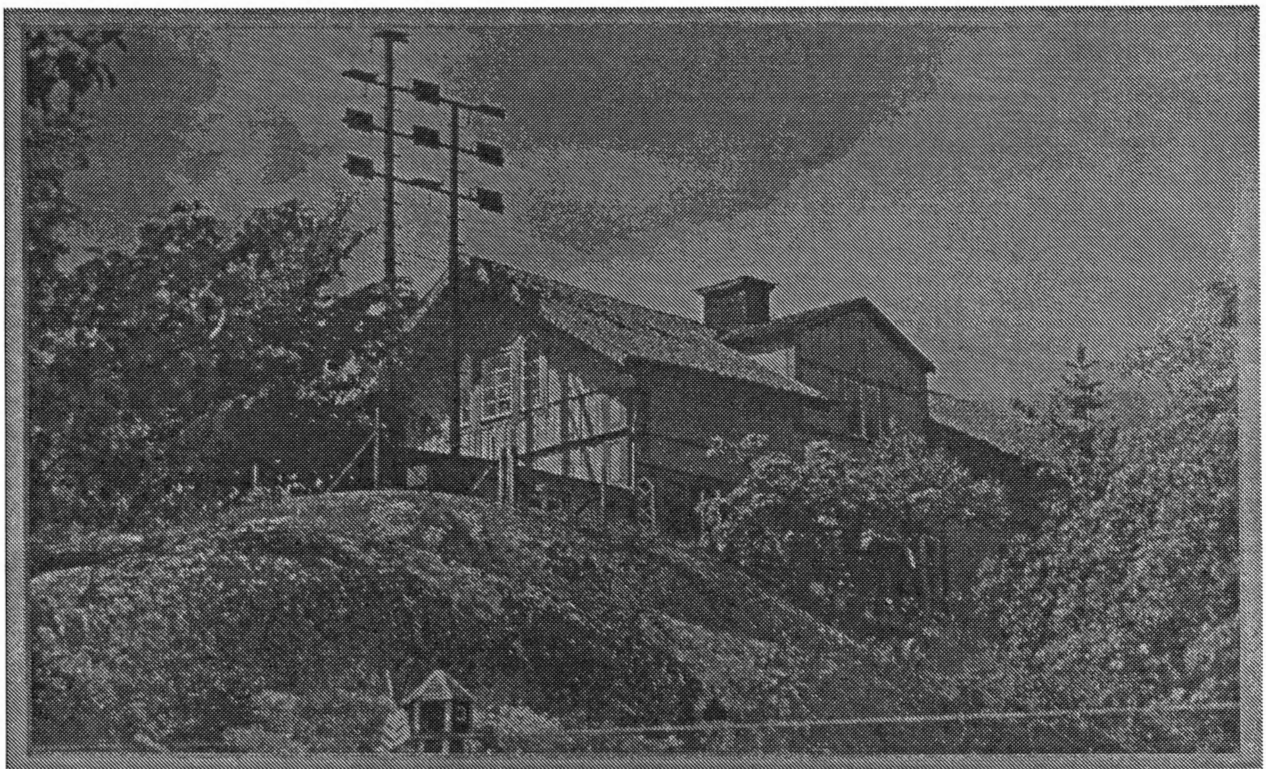


Figure 3.1: The Optic Telegraph

top. The flaps can be manipulated to produce 1024 different signals. Each signal represented a code which was interpreted in a code book, much like ASCII code. The code book included letters, numbers, common syllables, words, sentences and names. It also included control messages. The procedures also included an addressing scheme and schemes for encryption of messages.

3.2 Optic Fibres

The invention of optic fibres in recent years has brought a revolution in the telecommunications industry. The extremely high bandwidth of optic fibre allows a transmission speed of the order of 10 Gbps instead of

10-20 Mbps in coaxial cable. Later we shall see that even higher speeds are possible in optic fibres by means of *wave division multiplexing*. Bell Laboratories in the USA have managed to place 30,000 simultaneous phone calls on a single optic fibre.

3.3 Method of Transmission

Binary data is usually passed down an optic fibre in the form of a series of on/off pulses representing a digital bit stream. This differs from the electrical transmission media discussed in the previous chapter in that the transmitter information is carried in the form of a fluctuating beam of light in a glass fibre, as opposed to changes in voltage travelling down (usually) coaxial copper wire.

The overall pattern of transmission over an optic fibre cable is as follows: the electrical signal from the source is transmitted to a converter, either a laser or a light-emitting diode. These devices convert electrical signals into a series of light impulses. As these light impulses comprising the signal travel along the cable, the signal undergoes loss which is compensated for by repeaters placed on the fibre at 30-50 km intervals. At each repeater the signal is detected, converted to an electrical signal, amplified and re-emitted. When the optic signal reaches the end of the optic fibre, it is converted back into an electrical signal and transmitted to the destination.

3.4 Optic Media

Originally each signal was carried on a separate optic fibre which consisted of a narrow core surrounded by cladding; both of these layers are silicon. Protective plastic surrounds the core and cladding, preventing spurious light from entering the fibre. The core has a higher refractive index than the cladding, which, in practical terms, means that light rays travelling through the core at an angle less than a certain critical angle, are completely reflected by the cladding and will remain inside the core. A schematic diagramme of such a cable is shown in Figure 3.2 below. The angle q in Fig. 3.2 is called the Acceptance Angle. Any light

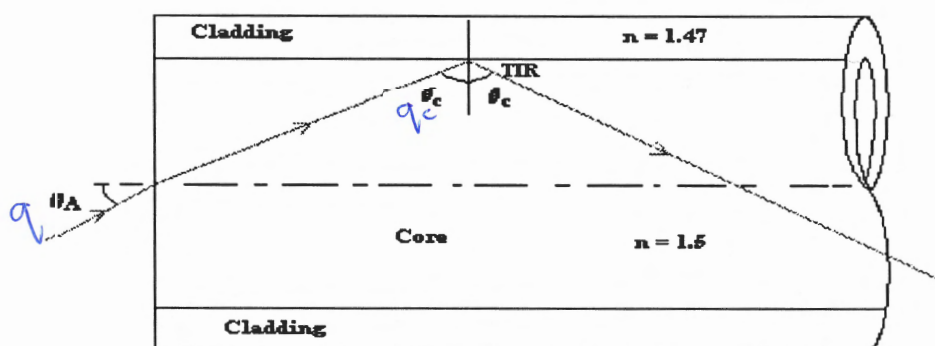


Figure 3.2: Principles and components of optic fibres

entering the fibre at less than this angle will meet the cladding at an angle greater than q_c . If light meets the inner surface of the cladding (the core - cladding interface) at greater than or equal to q_c then TIR occurs. So all the energy in the ray of light is reflected back into the core and none escapes into the cladding. The ray then crosses to the other side of the core and, because the fibre is more or less straight, the ray will meet

the cladding on the other side at an angle which again causes TIR. The ray is then reflected back across the core again and the same thing happens. In this way the light zig zags its way along the fibre. This means that the light will be transmitted to the end of the fibre.

In reality the light which enters the fibre is a focused beam, consisting of many millions of "rays" behaving in a similar way. They all zig zag along the core of the fibre, crossing over each other, and filling up the core with light. A pulse of light travelling along the core of the fibre is really a bundle of these rays.

In optic fibres, because the core is very thin, the distance between successive reflections is very small, and the optic fibre may be bent or twisted without affecting the transmission of the signal. In this way, light rays entering the core can be transmitted for long distances with a very low loss of energy.

3.4.1 Types of Fibre

There are currently two types of optic fibre in use: *step index fibre* both *multimode* and *single mode* and which is always *multi-mode*; both have advantages and disadvantages.

Step index fibre

Step index fibre is so called because the refractive index of the fibre "steps" up as we move from the cladding to the core of the fibre. Within the cladding the refractive index is constant, and within the core of the refractive index is constant as illustrated in Fig. 3.3(a).

Although it may seem from what we have said about total internal reflection that any ray of light can travel down the fibre, in fact, because of the wave nature of light, only certain ray directions can actually travel down the fibre. These are called the "Fibre Mode".

In a multimode fibre many different modes are supported by the fibre. The larger core of the *multimode fibre* enables light rays to enter the fibre at a larger angle than in the case of single-mode fibre. This means that individual rays of light can travel along many different paths or *modes* in a multimode fibre. Each impulse of light, encoding a single bit, consists of many such rays of light as shown in Fig. 3.3(b). Single mode fibre, on the other hand, because its core is so narrow, can support only one mode. This is called the "Lowest Order Mode" and Fig. 3.3(c) illustrates a single mode fibre. Single mode fibre has some advantages over multimode fibre which we will deal with later.

Graded index fibre

Graded Index Fibre has a different core structure from single mode and multimode fibre. Whereas in a step-index fibre the refractive index of the core is constant throughout the core, in a graded index fibre the value of the refractive index changes from the centre of the core onwards. In fact it has what we call a Quadratic Profile. This means that the refractive index of the core is proportional to the square of the distance from the centre of the fibre. This is shown in Fig. 3.3(d.)

Graded index fibre is actually a multimode fibre because it can support more than one fibre mode. But when we refer to "multimode" fibre we normally mean "step index multimode"

In **step-index multimode** fibre, there is a sharply defined interface between the core and the cladding (Figure 3.3(b)). This means that the different rays of light encoding the same bit will bounce off the cladding at different angles. Rays following a mode parallel to the axis of the fibre will travel a smaller distance than the rays which are reflected off the cladding more frequently. The rays thus arrive at the destination at

Modal dispersion - various modes/rays travelling different distances
 optic dispersion - various frequencies being delayed differently -
 chromatic dispersion?

3.4.1 Types of Fibre

47

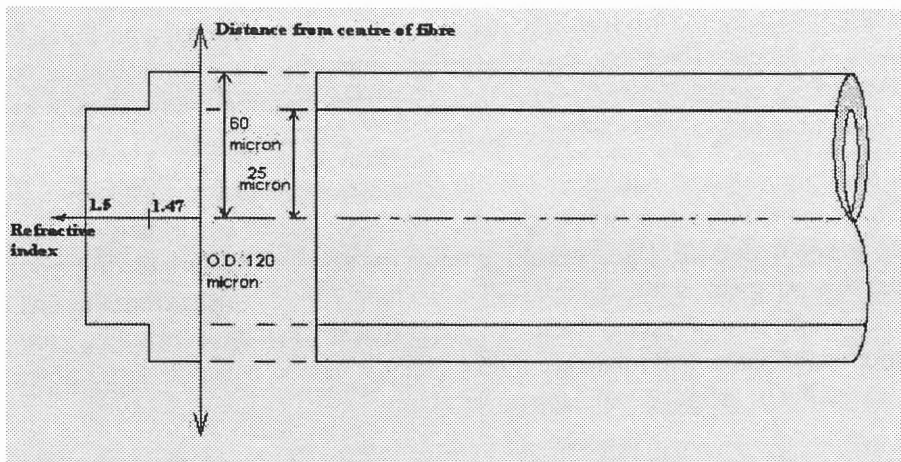


Figure 3.3: (a) Step index optic fibre

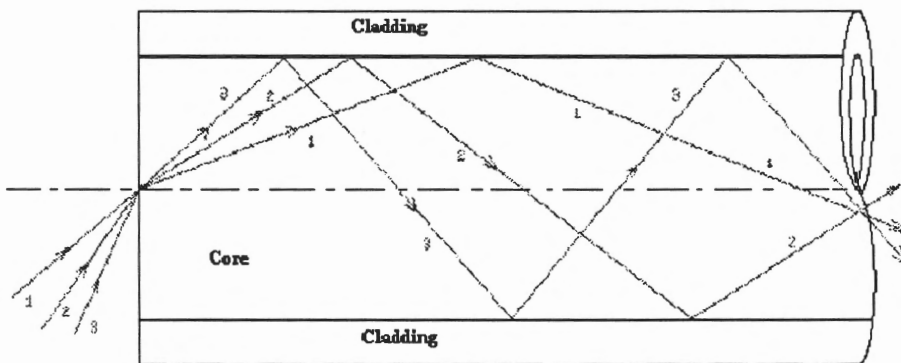


Figure 3.4: (b) Multimode fibre

mode = ray path

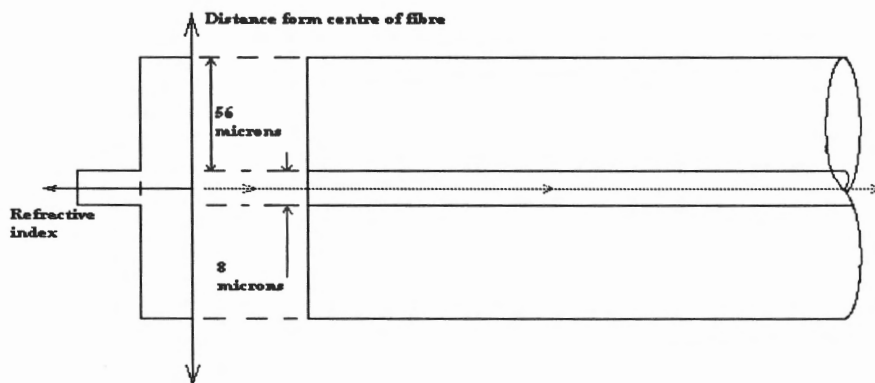
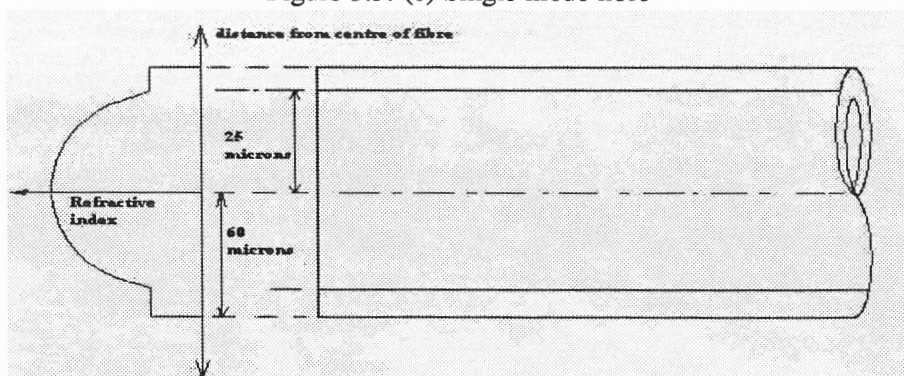


Figure 3.5: (c) Single mode fibre



maintains light coherence

Fig 3.5(d) Graded index fibre

slightly different times, causing the original light impulse to become dispersed, a phenomenon called *modal dispersion*. Because of modal dispersion, successive light impulses in multimode fibre can interfere with each other when travelling at high speeds or over long distances, limiting the bandwidth available through step-index multimode fibre to “only” 35 Mbps over 1km.

An improvement on the step-index multimode type of fibre is **graded-index multimode** fibre. These fibres do not have a sharply defined region between core and cladding, instead the refractive index of the changes gradually from the inside to outside. Light travelling down the axis experiences the highest optic index, and hence travels at the slowest speed, while rays travelling the longer modes away from the axis experience less refraction, thus travelling faster (Figure 3.3(b)). The time that it takes light travelling along each mode to reach the destination is the same, and relatively little modal dispersion occurs in these fibres. Graded-index multimode fibre is capable of carrying 500 Mbps over 1km.

Multimode fibres of both types are still in wide use, but **single-mode** fibres seem destined to supersede them. Single-mode fibres are similar to step-index multimode fibre in structure, but their much narrower core allows only one mode for the rays of light to travel along in the fibre, the *fundamental mode* (Figure 3.3(c)). Thus, in single-mode fibres, no modal dispersion occurs, allowing a much greater potential bandwidth in these fibres than in multimode fibres.

Single-mode fibres are far superior to both types of multimode fibre in the transmission distance possible, and also in the bandwidth available for exploitation. Graded multimode fibres do counter some of the problems of modal dispersion in multimode fibres, allowing better transmission characteristics, but these fibres are quite expensive to manufacture.

On the other hand, single-mode fibres are not perfect. There are two major problems with single-mode fibres, both due to their very narrow cores. Firstly, it is difficult to focus the transmission light beam exactly onto the core of a single-mode fibre without wasting some of the emitted light. Single-mode lasers must be used to achieve this, but these are more expensive than the light-emitting diodes (LEDs) or semi-conductor lasers that are used with multimode fibres. Secondly, also due to the narrow core, it is much more difficult to couple (join) two single-mode fibres to each other, than to couple multimode fibres. For these reasons, multimode fibres are still widely used for short or medium-distance transmission, while single-mode fibres are mostly used for long-distance or high-speed transmission.

3.4.2 Optic Signal Generation

The two main means of producing the light signals are light-emitting diodes (LEDs) and semi-conductor laser diodes (LDs).

LEDs produce light of a wide spectral width and are primarily used in multimode fibres which operate at wavelengths where chromatic dispersion is unimportant. However, they cannot be used efficiently with the high-speed single mode fibres. On the other hand, LDs couple easily with single-mode fibres, they allow wide bandwidth transmission, and they have a high output power. Nevertheless, the cheaper price and higher reliability of LEDs makes them a popular choice for multimode fibre transmission over short- to middle-distance applications.

The light impulses are intensity modulated by the LDs and LEDs. In more advanced LDs, the lasers are not physically switched on and off for each data bit, because this causes “jitter”, or interference, with the channel signal. In these LDs, the laser beam is continuously on, and the beam passes through two sheets of crossed polarised material, one of which is made of lithium niobate and has the electrical channel attached to it. When no electrical impulse is flowing down the channel, the two sheets of polarised material at right angles to each other prevent the passage of the laser beam into the channel. However, when an electrical

impulse stimulates the lithium niobate, its plane of polarisation changes, allowing light to pass through the two sheets and into the channel. A material with this property is called a *Pockels material*. Obviously, the faster that a change in polarisation can occur, the higher the potential transmission speed. A new Pockels material composed of organic molecules called *chromophores*, promises even better transmission feats than lithium niobate in the future.

At the receiving end of the optic fibre, the receptors need to be able to convert the light signal into an electrical signal extremely rapidly. The most popular systems used are the PIN photodiode¹. In PIN photodiodes the reverse-biased basic P-Intrinsic-N construction junction is exposed to light. Photons that exceed the bandgap energy of the intrinsic material generate electron-hole pairs that generate external current.

The avalanche photodiode is used mainly in high-speed, long-haul systems. The avalanche photodiode is highly sensitive to light, which excites the electrons, producing a voltage. The material used in these photodiodes causes an "avalanche" effect, magnifying the voltage produced in the electrical channel.

3.5 Signal Loss in Optic Fibre

Despite the very great bandwidth available for light transmission, losses in the signal still occur. The three main causes are *modal dispersion*, occurring only in multimode fibre, *chromatic dispersion*, and *fibre or optic loss*. Modal dispersion and chromatic dispersion are analogous to the attenuation and delay distortion present on electrical channels, while fibre loss is analogous to noise distortion.

1. Modal dispersion in multimode fibres is due to different rays of light travelling along different paths in the core, causing dispersion in an impulse; it has been discussed in conjunction with multimode fibres and single-mode fibres in the previous section.
2. Chromatic dispersion occurs in all fibres because each light impulse is composed of a number of wavelengths. The problem with these different wavelengths of light is that each travels at a different speed through the medium. The signal thus becomes dispersed, in a manner analogous of modal dispersion. Some lasers can produce a signal of a single wavelength, but such lasers are incredibly expensive. Multimode fibres are particularly guilty of chromatic dispersion because their light sources (especially LEDs) transmit on a relatively broad band of wavelengths. Single-mode fibres are not exempt from this dispersion either because they transmit pulses containing the Fourier components of different wavelengths.

There are two windows of low optic loss: one lies around 1.3 μm and the other, much lower level, is around 1.55 μm . The development of new silicon fibres with zero chromatic dispersion at 1.55 μm has led to this wavelength becoming the new standard wavelength of optic transmission.

3. A third cause of signal loss is termed optic loss and is due to very slight impurities in the glass of the core and cladding. These impurities cause the fibre to absorb the light impulses. This weakens the signal and is the reason for the repeaters on optic fibre lines.

The above effects have meant that repeaters (or boosters) had to be installed in any optic transmission path to ensure error free transmission. Recent technology however, where the fibre is doped with erbium neobate, the optic signal is boosted along as it travels through the medium, thus eliminating the need for signal regenerators almost completely.

¹The term "PI" is an acronym for P-Intrinsic-N, where a layer of intrinsic semiconductor material is sandwiched between p- and n-semiconductor material

3.6 Advantages and Disadvantages of Optic Fibres

The future of optic fibres is very “bright” for a number of reasons:

1. The light passing through the fibre is made up of different frequencies of wavelengths. Light waves give rise to a much higher capacity than electrical signals (cf 10-20 Mbps for coaxial cable over a short distance) and hence optic fibre cable can be used for transmitting very high bit rates, in the order of hundreds of megabits per second.²
2. Furthermore, the use of a light beam makes optic fibre cable immune to the effects caused by spurious electromagnetic interference signals and crosstalk effects. Optic fibre cable, therefore, is also useful for the transmission of lower bit rate signals through noisy electrical environments – in steel plants for example, which employ much high-voltage and current-switching equipment. This had lead to plans by British Rail to lay fibre optic cable alongside its railway lines.
3. Optic signals travel much further without attenuation, so repeaters are needed only every 30-50 Km compared to 5 Km for copper cables.
4. It is also being used increasingly in environments that demand a high level of security, since it is difficult to physically tap an optic fibre cable.
5. The error rate in fibre is very low, only 10^{-9} errors per bit compared to 10^{-6} typical of copper cables.
6. Optic cables are small and light-weight. For example, 900 copper wire pairs stretched across 300 m weighs just over two tonne. Two optic fibres pulled along the same length weighs approximately 40 kg, including their protective coverings.

On the other hand, fibre optic cable has a few disadvantages:

1. Installation costs are much greater than for electrical transmission media, because the fibres are physically weak and difficult to couple together. Losses in the signal may occur when the fibres are coupled, so it is difficult to reconfigure the system for changing needs.
2. Fibre is primarily a point-to-point medium which does not allow for bus protocols such as Ethernet.

For these reasons, the usage of optic fibre cable is currently limited to long-distance high-speed uses, and environments with a lot of electrical noise (e.g. a factory with a lot of machinery). However, as the demand for greater bandwidth shows no sign of abating, fibre optic cable is definitely the network transmission medium of the future.

3.7 Exercises

Exercise 3.1 *Name the advantages and disadvantages of optic fibre.*

Exercise 3.2 *Different from radio signals, optic fibre, like copper have to run physically from one point to the next. Can you think of ways other than digging a trunk through the fields to lay the optic fibre.*

²The theoretical limit is in the order of 1 Tbps (Terrabits per second) without multiplexing

Chapter 4

Wireless Communication

4.1 Objectives of this Chapter

Once familiar with the material in this chapter the reader should

1. know what radio signals are and the
2. use of the available radio frequency spectrum, as well
3. as have a basic understanding about satellite communication and the various kinds of satellite systems (GEO, LEO and MED) including
4. the advantages and disadvantages of each.

Ever since man has been able to speak and communicate he wanted to be able to transmit messages over long distances without having to have a physical connection. In the most basic form this includes techniques such as smoke signals of early tribes, heliographs that reflect the sunlight or lanterns used to transmit morse code on ships. But yet this was not enough and in the 20th century, with its rapid advances in communications technology and electronics more was demanded.

When electrons move, they create electro magnetic waves that can propagate through free space. These waves were first predicted by the British physicist James Clerk Maxwell in 1865 and first produced and observed by the German physicist Heinrich Herz in 1887. By attaching an antenna of the appropriate size (around a wavelength in length) to an electrical circuit, the electromagnetic waves can be broadcast efficiently and received by a remote receiver.

Interestingly enough, the dawn of wireless communications actually has its roots set back in the end of the 19th century with an Italian called Guglielmo Marconi. It was his experiments that demonstrated that a continuous wave could be propagated through the atmosphere or *ether*, and over the great boundary at the time, the horizon. The problem was that the earth's surface is curved and so a direct line-of-site is only possible for several kilometres and the further the sender is from the receiver, they will lose line-of-sight as they dip beneath the horizon. Marconi showed that radio waves followed the earth's curvature, by bouncing off the upper part of the atmosphere known as the ionosphere. He successfully transmitted a faint signal from Cornwall, England to his receiving apparatus in Newfoundland, Canada.

4.2 Frequency Spectrum

Wireless communication refers to just that; sending electro magnetic signals without wires. As such the term applies to a signal sent at any frequency. The frequency spectrum, although wide, is still finite. Clearly the spectrum should be regulated nationally and internationally to prevent chaos. As argued above, the wider the frequency band allocated the higher the data rates. In the USA the Federal Communications Commission (FCC) decides who gets what whereas, in the rest of the World, the ITU-R is responsible for the allocation of frequencies. The electro magnetic spectrum and its allocations is illustrated in Table 4.1. The radio, microwave, infrared and visible light bands can all be used to transmit digital signals by modulation of the waves. Ultraviolet light and beyond would be even better due to the higher frequencies, but they are hard to modulate using conventional techniques and do not propagate through objects such as buildings or other structures. High frequencies are also not very good for living things as anyone who owns a microwave oven knows. At the other extreme, below 10⁴Hz we have the audio signal range.

Starting at 10 KHz, the names and frequency ranges listed in the figure are from the ITU standards for the various wave bands. In this way LF, MF, and HF refer to Low, Medium and High Frequency; the higher bands are similarly named Very, Ultra, Super, Extremely and Tremendously [*seriously*] High Frequency. By contrast visible light has a frequency of just over 100THz (10¹⁴) and optical signal frequencies lie in the frequency range 10¹⁴Hz to 10¹⁵Hz.

In vacuum all electromagnetic waves travel at the same speed, no matter what their frequency. This speed, denoted c is the speed of light or approximately 3×10^8 meters per second. At that speed a signal travels about 30 centimetres in a nanosecond. In copper or optical fibre the speed slows to about two-thirds of the value and is somewhat frequency dependent.

The fundamental relationship between the frequency f , the wavelength λ and the speed of light (in a vacuum) is

$$(4.1) \quad \lambda f = c.$$

$$c \approx 3 \times 10^8 \text{ m/s}$$

Amplitude
Frequency
Phase
Minimum - combination of frequency and phase
shift keying

Frequency Band Name	Upper Limit	Modulation	Highest Data Rate	Typical Applications
Low (LF)	300KHz	ASK, FSK, MSK	100bps	Telephone, Twisted pair wires
Medium (MF)	3MHz	ASK, FSK, MSK	1Kbps	Commercial AM radio, coaxial cable
High (HF)	30MHz	ASK, FSK, MSK	3KBps	Shortwave radio, Television signals
Very High (VHF)	300MHz	FSK, PSK	100KBps	VHF Television, FM Radio
Ultra High (UHF)	3GHz	PSK	10MBps	UHF Television, Terrestrial Microwave
Super High (SHF)	30GHz	PSK	100MBps	Terrestrial and Satellite Microwave
Extremely High (EHF)	300GHz	PSK	750MBps	Terrestrial Microwave
Tremendously High (THF)	3THz	PSK	15GBps	Not Known

Table 4.1: Electro magnetic spectrum and its uses

Since c is a constant, if we know f we can find λ and vice versa. If we solve Eq. 4.3 for f and differentiate with respect to λ we get

$$(4.2) \quad \frac{df}{d\lambda} = -\frac{c}{\lambda^2}.$$

Using infinitesimal differences rather than differentials and ignoring signs, we obtain

$$(4.3) \quad \Delta f = \frac{c \Delta \lambda}{\lambda^2}.$$

Thus, given the width $\Delta \lambda$ of the wave band, we can compute the width Δf of the corresponding frequency band. From the section on Fourier analysis we know that the wider the bandwidth the higher the number of harmonics and the higher the capacity or data rate. The Ku-band frequency range for satellite transmission is 11.7 to 12.2GHz or $\Delta f = .5 \times 10^{12}$ allowing for a capacity of easily 50Mbps, assuming that roughly 10 harmonics will allow for a good quality signal using a crude on-off coding technique. In reality coding techniques which give rise to a multiple of this rate are used.

4.3 Radio Transmission

Radio transmission refers to any wireless technique that uses *radio frequencies (RF)*, i.e., in the 10KHz to 900MHz range to transmit data. Cellular telephones, for example, generally operate in the 824 – 849MHz, 869 – 895MHz and 1800MHz range. Cordless telephones and wireless LANs (see Chapter 8) use the 902 – 928MHz range whereas the 2.4 – 2.5GHz and 5.8 – 5.9GHz ranges are unregulated.

The properties of radio waves are frequency dependent. At low frequencies, radio waves easily pass through obstacles such as buildings, but the power falls off sharply with distance from the source, approximately $1/r^3$ for distance r from the sender. At higher frequencies radio waves tend to travel in straight lines and be deflected by obstacles as any cellphone user knows. In general radio transmission is making a comeback largely through the success of cellular networks and the planned *Bluetooth* technology which is intended to link various mobile devices (telephone, laptop, Personal Digital Assistants (PDAs)) up to a range of 10 meters in a Personal Area Network (PAN).

A major application of Bluetooth technology is ability to electronically pay parking meters, for bus tickets, at shop checkout points and so on through the use of Bluetooth enabled devices – or so the theory goes. The primary members of the Bluetooth consortium are 3Com (inventors of the Ethernet), Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba.

In Chapter 14 cellular communication networks which are increasingly important for their improved data carrying capacity, will be discussed in more detail.

4.4 Terrestrial Microwave Signals

Microwave covers part of the UHF and all of the SHF band. Because of their high frequencies, microwave signals are easily attenuated and deflected by obstacles so that a microwave transmitter and receiver require a clear line of sight. The frequencies are in the range 3 to 30GHz and the higher the frequency the higher the potential bandwidth and therefore the higher the potential capacity (see Eq. 4.3).

The most common type of microwave antenna is the parabolic dish which can be up to 3 meters in diameter. With no intervening obstacles, the maximum distance between two microwave antennae conforms to the formula

$$(4.4) \quad d = 7.14\sqrt{Kh}$$

where d is the distance between antennae in kilometers, h the high of the antennae in meters, and K is an adjustment factor to account for the fact that microwaves are bent with the curvature of the earth and will hence propagate further than the line of sight. The empirical value of $K = 4/3$. Two microwave antennae at a height of 100 meters can thus be as far as $7.14 \times \sqrt{133} = 82$ kilometers apart. To achieve transmission over longer distances, a series of microwave relay towers are used as can frequently be observed along the major highways in the country where it carries voice, television and data signals.

As with any transmission system, a main source of loss is attenuation. For microwave and radio frequencies the loss is given by the formula

$$(4.5) \quad L = 10\log \left(\frac{4\pi d}{\lambda} \right)^2 \text{ dB}$$

where d is the distance and λ is the wavelength in the same units. Note that loss varies with the square of the distance whereas in copper conductors it is linear. Attenuation is increased with rainfall (and flying ducks) especially at frequencies above 10GHz.

Another source of transmission impairment is interference between adjacent signals. With the growing popularity of microwave, transmission areas overlap and interference is always a danger which is why the frequency bands have to be strictly regulated.

In general a microwave link is less costly to install than cable for moderate distances, has a high data rate for as wireless transmission goes, requires little to no maintenance and is fairly easy to implement. The drawbacks we have mentioned already.

4.5 Satellite Communication

A communication satellite is in fact nothing more than a microwave relay station linking two or more earth-based microwave transmitters/receivers, known as earth or ground stations. The satellite receives signals on one frequency band (called the "uplink" and amplifies it and transmits them down to earth on a different frequency called the "downlink" as illustrated in Fig. 4.1. This is necessary to avoid interference between the signals. A single orbiting satellite will operate on a number of bands called *transponder channels*.

The bandwidth capability of satellite systems varies and depends on the frequency at which the satellite transmits. Four common frequencies are

- C-band with a 4-GHz¹ downlink and a 6 GHz uplink (the uplink frequency is always higher than the downlink).

¹Hz is an abbreviation for "Hertz" which is another name for cycles per second

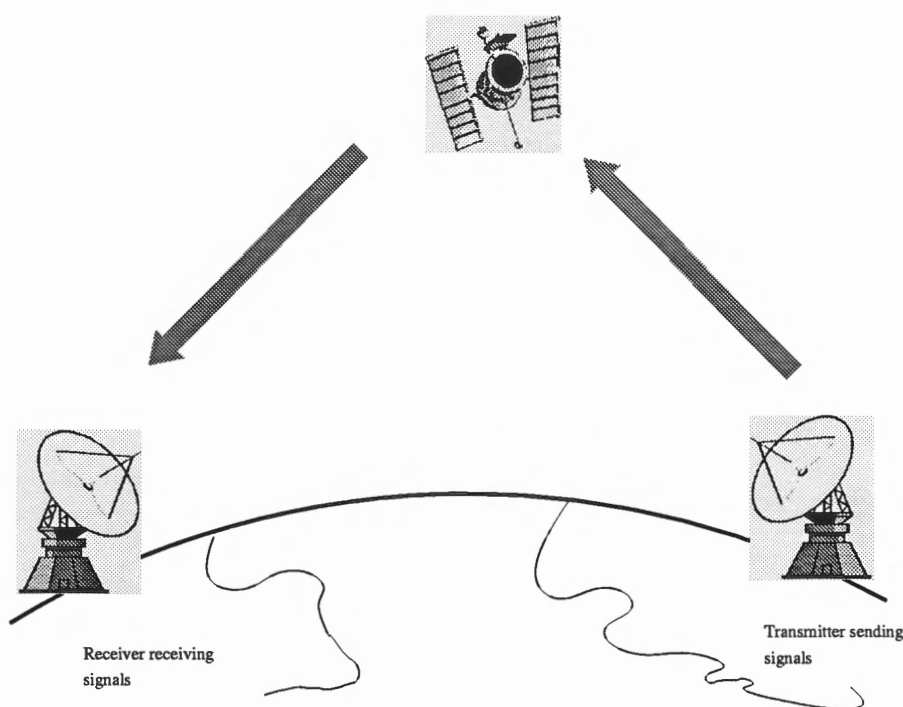


Figure 4.1: Satellite communication

- *Ku*-band with 14-GHz up and 12-GHz down.
- *Ka*-band with a 28-GHz uplink and a 18-GHz downlink.
- *V*-band is above 30-GHz and is still under development.

In order for a communication satellite to work properly, it is generally required to remain stationary with respect to an observer from earth, i.e., the sending/transmitting station. According to Kepler's law, the orbital period of a satellite varies as the orbital radius to the $3/2$ power. At an altitude of 36,000 km, the orbital period is 24 hours, so it rotates at the same rate as the earth under it. An observer looking at a satellite in a circular *equatorial* (note) orbit sees the satellite hanging in a fixed spot in the sky, apparently fixed in the sky. This is essential, since otherwise an expensive tracking antenna would be needed. In fact the satellite does not remain completely stationary but gently moves around a cubic area of about 50 Km in space. Such a satellite is said to be in a *geosynchronous* orbit and the satellites are referred to as *Geostationary Earth Orbit* (GEO) satellites. To prevent interference GEO satellites cannot be placed any closer than 4° apart at the same altitude. In other words no more than 90 satellites can be parked on the equatorial orbit. In fact however, because they are so high up, only 8 GEO satellites are needed to cover the entire world.

Probably the main disadvantage of satellite communication is the distance involved. Although microwave signals travel at 3×10^8 meters per second, it still requires about 125 millisecond to reach the satellite with a round trip delay of a quarter of a second. At high transmission speeds 250 milliseconds is a lifetime compared to about 3 milliseconds per kilometer for terrestrial microwave links.

In order to avoid the problem with delay several commercial satellite projects are being developed for the sole purpose of data communication. One such project is the *Teledesic network* project whose main investor is the Microsoft company. The Teledesic project will comprise 840 interlinked Low Earth Orbit (LEO) satellites that are to provide a digital skin to the earth for access to voice, computer and video

4 ways of sharing a common transmission medium in a technique called "multiplexing"

- time division multiplexing (TDM)
- space division multiplexing (several fibres)
- wave, or frequency division multiplexing (FDM, WDM)

56- code division multiplexing

CHAPTER 4. WIRELESS COMMUNICATION

communications beginning in 2002. It is intended to support 1 million full-duplex (bi-directional) E1-lines (see Chapter 6) each of capacity 2.048 Mbps. Compared to GEO satellite, LEO satellites orbit anywhere anywhere from 500 to 2000 Km above the earth. At the speeds at which they need to travel to stay aloft, LEO satellites are only in sight of a terrestrial antenna for at most 15 minutes. Consequently, a LEO-based satellite channel must be passed from one satellite to the next in the constellation. The propagation delay on LEO satellites ranges from 40 to 450 milliseconds making them better equipped to support real time applications such as video conferencing.

There are also so-called Medium Earth Orbit (or MEO) satellite systems which, as the name indicates, orbit from 9 000 to 20 000 Km away and have a propagation delay of at most 250 milliseconds. About 20 MEO satellites will cover the surface of the globe.

To summarize, the following are the characteristics of a typical satellite:

- A single satellite is visible to roughly a quarter of the earth's surface.
- Within that area (sometimes called the *footprint* of the satellite), the cost of transmission is independent of the distance.
- Network connection can be affected by simply pointing an antenna at the satellite.
- Very wide band widths are possible.

An interesting property of satellite broadcasting is precisely that: it is broadcasting. All stations under the downward beam can receive the transmission, including "pirate stations" the common carrier may know nothing about. The implications for privacy are obvious. Also, because a user can listen to the retransmission of his own data, he can tell whether it is in error or not. This property has important implications for error handling strategies.

It is important to know that a single optical fiber, costing a few cents, has more bandwidth than all the satellites ever launched. Whereas satellite networks were originally considered to be the only means of long range communication it is no longer the case. Inter-continental optical fibre cables are becoming more and more common, are cheaper, and more reliable and secure. Satellite links are still essential, however, for communication with moving ground stations such as ships at sea.

4.6 Exercises

Exercise 4.1 List the advantages and disadvantages of terrestrial microwave communication.

Exercise 4.2 The following two alternatives are available for a point-to-point communication link.

1. A GEO-satellite link with a 10Mbps capacity
2. A terrestrial digital communication link with 2.048Mbps capacity.

Assume that after a message has been sent it needs to be acknowledged before the next one can be sent. If the terrestrial link has a maximum message size of 1Kbs, how many bits must a message on the satellite link have for it to have a lower delivery time?

Chapter 5

Channels and their Properties

5.1 Objectives of this Chapter

There are several properties and concepts which are common to all communication links regardless of the medium. In this chapter we shall learn about

- The directions of communication that are possible on a communication channel, called simplex half-duplex and full-duplex.
- Ways of synchronising the sender and receiver to ensure that they read and interpret the same bits.
- Individual network users do not always require a constant capacity all of the time. One can use this fact to have several users share a common medium in a technique called *multiplexing*. Details of Synchronous Time Division Multiplexing (STDM), Statistical TDM and Wave (or Frequency) Division Multiplexing are important for a full understanding of a communication channel. This includes the recent variations of Digital Subscriber Line (xDSL) which have been proposed or are being rolled out by telecommunication companies (telcos).
- Finally we discuss the differences between circuit and packet switching and mention cell switching which is a variation of packet switching used in Asynchronous Transmission Mode (ATM) networks, but which is not quite the same as packet switching.

5.2 Introduction

There are several properties and concepts which are common to all communication links regardless of the medium. In this chapter we shall learn about these.

5.3 Simplex, Half-duplex and Full-Duplex Communication

A distinguishing feature of any data communication path is the direction of data transfer between two communicating systems. Data units may (see Figure 5.1):

1. travel in *one* direction only from one system to the other, called **simplex** communication.
2. travel in *both* directions, but not simultaneously, called **half-duplex** communication e.g.. two-way radios
3. travel in *both* directions *simultaneously*, called **full duplex** communication

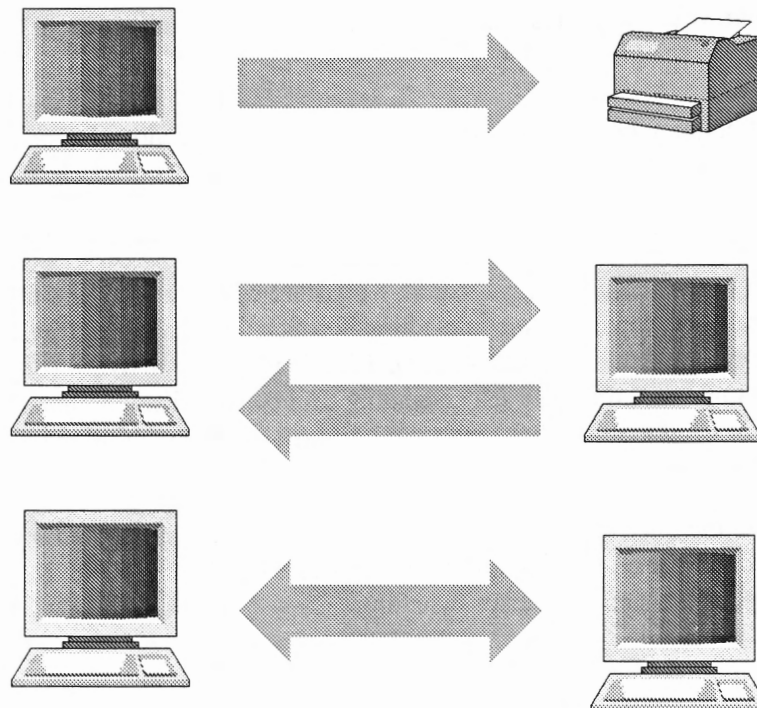


Figure 5.1: Simplex, Half-duplex and Full-duplex Communication

Two-way communication is complex, especially over interconnected networks. Protocols must be used to make sure information is received correctly and in an orderly manner. This will be discussed in later chapters.

5.4 Asynchronous and Synchronous Transmission

In order for computers and terminals to communicate, they must first notify each other that they are about to transmit. Second, once they have begun the communication process, they must provide a method to keep

both devices aware of the ongoing transmissions i.e. a transmitter, such as a terminal must transmit its signal so that the receiving device knows when to search for and when to recognize the data as it arrives. This requirement means that a mutual time base, or a common clock, is necessary between the receiving and transmitting devices.

The transmitting machine first forwards to the receiving machine an indication that it is sending data. If the transmitter sends the bits across the channel without prior notice, the receiver will likely not have sufficient time to adjust itself to the incoming bit stream. In such an event the first few bits would be lost, perhaps rendering the entire transmission useless.

This process is part of the communications protocol and is generally referred to as *synchronization*. As the clocking signal on a line between two machines changes, it notifies the receiving device to examine the data line at specific times. Clocking signals perform 2 valuable functions:

- they synchronize the receiver onto the transmission before the data actually arrives
- they keep the receiver synchronized with incoming bits.

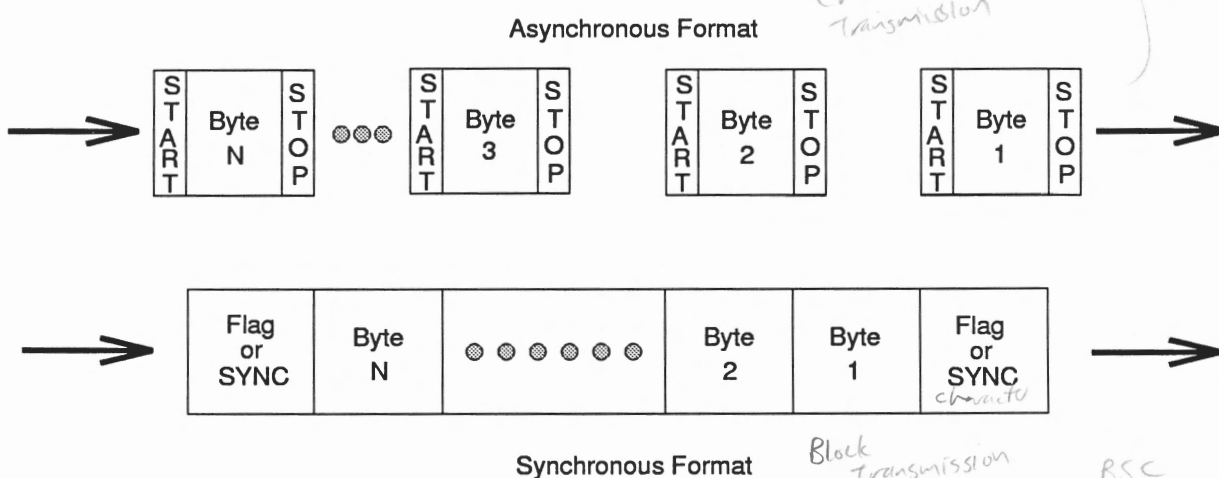


Figure 5.2: Asynchronous and Synchronous Transmissions

Two data formatting conventions are used to help achieve synchronization. These two methods are shown in Fig 5.2. The first approach is called asynchronous formatting. With this approach each character has start and stop bits (i.e. synchronizing signals) placed around it. The purpose of these signal is to firstly alert the receiver that data is arriving and secondly, to give the receiver sufficient time to perform certain timing functions before the next character arrives.

A more sophisticated process is synchronous transmission which uses a self-clocking bit-synchronisation mechanism. Synchronous formats eliminate the intermittent start/stop signals around each character and provide signals which precede and sometimes follow the data stream. The preliminary signals are called sync bytes or flags and their principal function is to alert the receiver of incoming user data. This process is called *framing*.

Synchronization takes place at *different logical levels*, and unless this is understood properly some confusion may arise later when Data Link Protocols are discussed. To start with, it is important to note that for phase-coherent analog transmission, the demodulator requires *carrier synchronism*, that is, a precise knowledge of the transmitted carrier signal phase and frequency.

Apart from this it is also necessary to maintain *bit synchronism*, by which is understood the receiver's ability to receive a stream of bits from an incoming *analog signal* by sampling for the bits at the proper times. The receiver needs to know when to start sampling the first bit, and the bit duration between consecutive bits.

In most cases, the transmitter/receiver ¹ is a *synchronous modem*, which will recover symbol timing and provide bit timing in a separately received data timing signal. It is also possible that no bit synchronization is provided by the modem, in which case, not surprisingly, it is referred to as an *asynchronous modem*.

Such modems contain no inherent clocking mechanism and are unaware of the bit rate. The receiving modem turns analog signals into digital bits, but leaves the identification of the bits to the so-called *channel adaptor*. The modem will work at any bit rate up to a certain maximum, and the data will be successfully received if the two channel adaptors are set to the same rate.

When the receiver and transmitter are in close proximity, binary signals can be transmitted directly between receiver and transmitter. In that case bit synchronism can be maintained by

- a *selfclocking binary signal* such as the Manchester code discussed in Sec. 2.5.1 on page 34 where a signal transition occurs at the beginning of every symbol (bit) period, or
- by a single shared clock for both the transmitter and receiver.

These methods for achieving bit synchronism between sender and receiver determines two basic forms of transmission: *asynchronous and synchronous*. In an *asynchronous* transmission system, bit synchronism is in fact maintained, but only when data transmission is taking place and not during idle periods. The amount of data transmitted at a time consists of a *single character* (5 to 8 bits). In a synchronous transmission system, bit synchronism is maintained at all times. Data Link Control protocols (DLC) ² normally used with a synchronous transmission system are said to be *synchronous protocols*, (and the DTE's using them are called *synchronous devices*), while DLC protocols normally used with an asynchronous transmission system are said to be asynchronous or *stop-start* protocols. We note that it is possible to use character-oriented protocols for both synchronous and asynchronous transmission systems. One example is the American National Standards Institute (ANSI) X3.28 protocol standard, which is similar to the character oriented Binary Synchronous Control (BSC) protocol described in Chapter 7. However, both are much more sophisticated than the early start-stop protocols and we will classify them as synchronous protocols.

5.5 Multiplexing Signals

Multiplexing refers to the simultaneous transmission of several messages down a single channel. Multiplexing is most frequently associated with optical signals; we present here a general description of multiplexing however which applies to any signal.

Fig. 5.3 explains multiplexing in its simplest form. The multiplexor combines (multiplexes) data from n slow input lines and transmits over a high capacity link to the a demultiplexor. The demultiplexor reverses the process in that it separates (demultiplexes) the single data stream into its n components and delivers them to the appropriate output line.

Example 5.1 Consider a terminal network. With four terminal Time Division Multiplexing (TDM), each terminal is allocated one-fourth of the output time slots, regardless of how busy it is. If each of the terminals operates at 1200 bps, the output line must be $4 \times 1200 = 4800$ bps, since four characters must be sent during each polling cycle.

¹ Also called a "transceiver"

² Discussed in Chapter 7

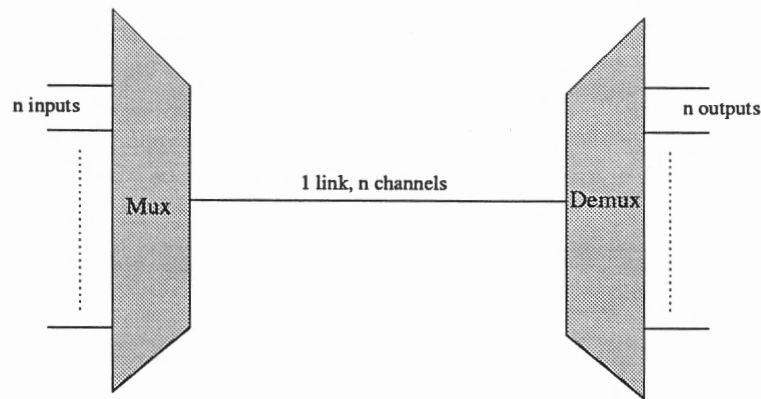


Figure 5.3: Multiplexing

In the case of fibre optic channels most long distance fibre optic transmission links contain many optical fibres. In exceptional cases, each fibre carries only *one* message, for example a single telephone call. In most cases, however, hundreds of messages are transmitted down the same optical fibre. The two most widely used types of multiplexing optical signals are *Time Division Multiplexing* (TDM) and *Wavelength Division Multiplexing* (WDM) neither of which is however limited to optical signals.

5.5.1 Synchronous Time Division Multiplexing (TDM)

Time Division Multiplexing involves the interleaving of bits, bytes or blocks of data from several channels into one channel, all travelling at the same carrier frequency. It provides the user with the full channel capacity but divides the channel usage into time slots. Fig. 5.4 illustrates the structure of a typical STDM

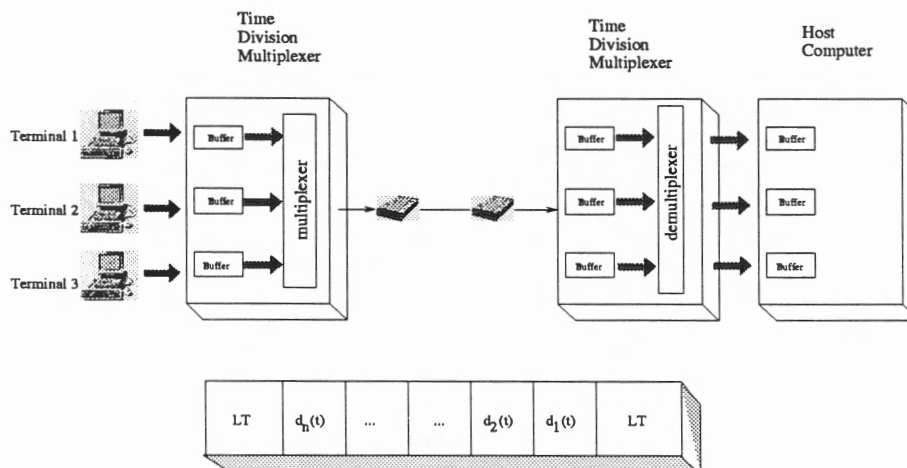


Figure 5.4: Synchronous Time Division Multiplexing

system. The TDM scans each device and organises the data into a frame. The slots in the frame are fixed such that $\sum_i d_i(t) = d_0(t)$ where $d_i(t)$ is the capacity of each device in bits/s; $d_0(t)$ = capacity of the output line in bps; N is the number of input devices.

The big disadvantage of TDM is that when an input device (eg. a terminal) has no traffic, an output time slot is wasted. The output slots are filled in strict rotation, as in Fig. 5.4. If there are no data, dummy characters

are used. It is not possible to skip a time slot, because the receiving end keeps track of which character came from which terminal by its position in the output stream. Initially, the multiplexer and the computer synchronize themselves. Both know that the order to be used, for example, is 012301230123 ... The data themselves carry no identification of their origin. If the multiplexer skipped a time slot when there were no data from a sender, the receiver would get out of phase and interpret the origin of succeeding characters incorrectly.

5.5.2 Statistical Time Division Multiplexing (STDM)

Statistical multiplexors were developed to obviate the waste of empty slots of TDM. Vacant slots occur when an idle terminal has nothing to transmit in its slot. Statistical TDM multiplexers (STDMs) dynamically allocate the time slots among active terminals. Fig. 5.5 below shows how the STDM “compresses” the empty slots out of the signal. It is for this reason that STDMs are also called *concentrators*.

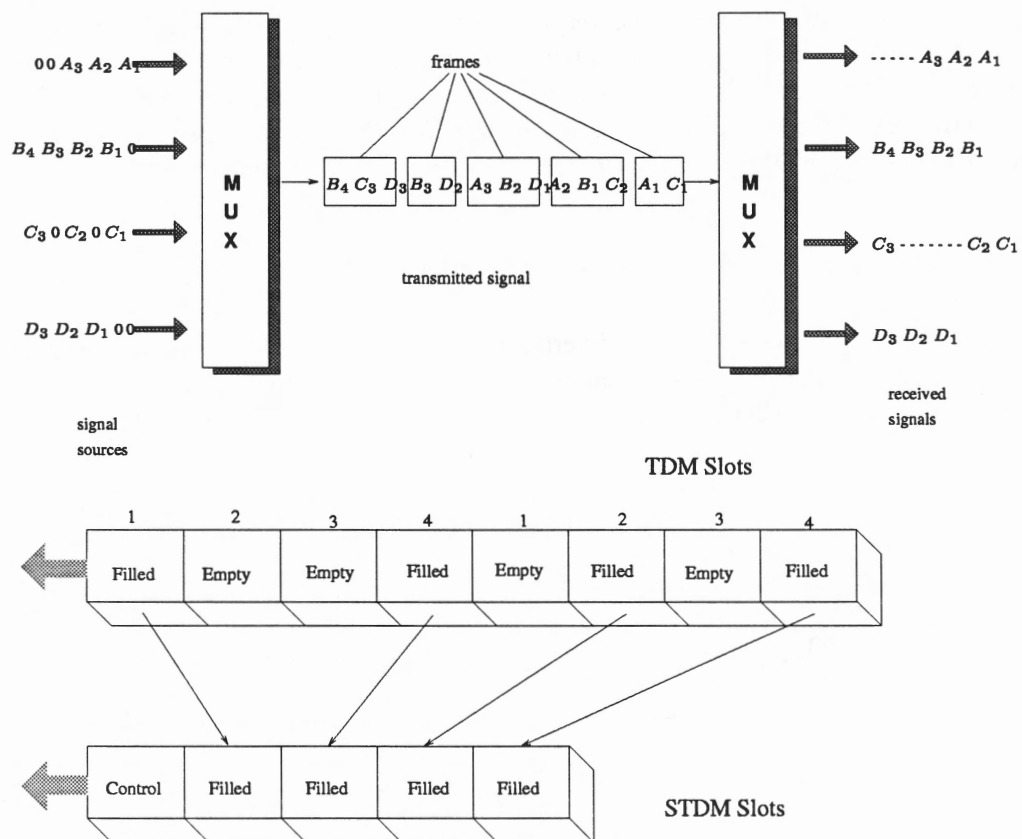


Figure 5.5: Statistical Time Division Multiplexing

Fig. 5.5 illustrates the introduction of a control field in the STDM frame. It is used to identify the owners of the slots. The demultiplexer uses this control field to reassemble each channel's traffic at the receiving node and pass it on to the proper recipient.

Unfortunately, concentration has an inherent difficulty. If each terminal suddenly starts outputting data at its maximum rate, there will be insufficient capacity on the output line to handle it all. Some data may be lost. For this reason, concentrators are always provided with extra buffers in order to survive short data surges. The more memory a concentrator has, the more it costs but the less likely it is to lose data. Choosing the

in STDM, every block of data has to have a HEADER identifying it.

appropriate parameters for the output line bandwidth and concentrator memory size involve trade-offs. If either is too small, data may be lost. If either is too large, the whole arrangement may be unnecessarily expensive. Furthermore, the optimum choices depend on the traffic statistics, which are not always known at system design time.

In the case of optical signals is that although the optical channel can happily carry large capacity, current electronic equipment cannot perform the multiplexing and demultiplexing at fast enough rates. For this reason, purely optical equipment is being developed to perform these functions, ensuring that the electrical components only have to cope with the signal when it has been demultiplexed to its original data transmission rate. The TDM process is then called Optical Time Division Multiplexing (OTDM).

5.5.3 Wavelength Division Multiplexing (WDM or FDM)

cannot have full duplex communication with physical transmission medium

Wavelength (or Frequency) Division Multiplexing, illustrated in Fig. 5.6, does not involve a faster changing signal at the same wavelength. Instead, several signals of different wavelengths are sent down a single fibre or channel simultaneously. This approach in effect divides the transmission frequency range (the bandwidth) into narrower bands (called subchannels). The subchannels are lower-frequency bands and each band is capable of carrying a separate voice or data signal. Consequently, FDM is used in a variety of applications such as telephone systems, radio systems, and cable television. WDM allows each signal sent

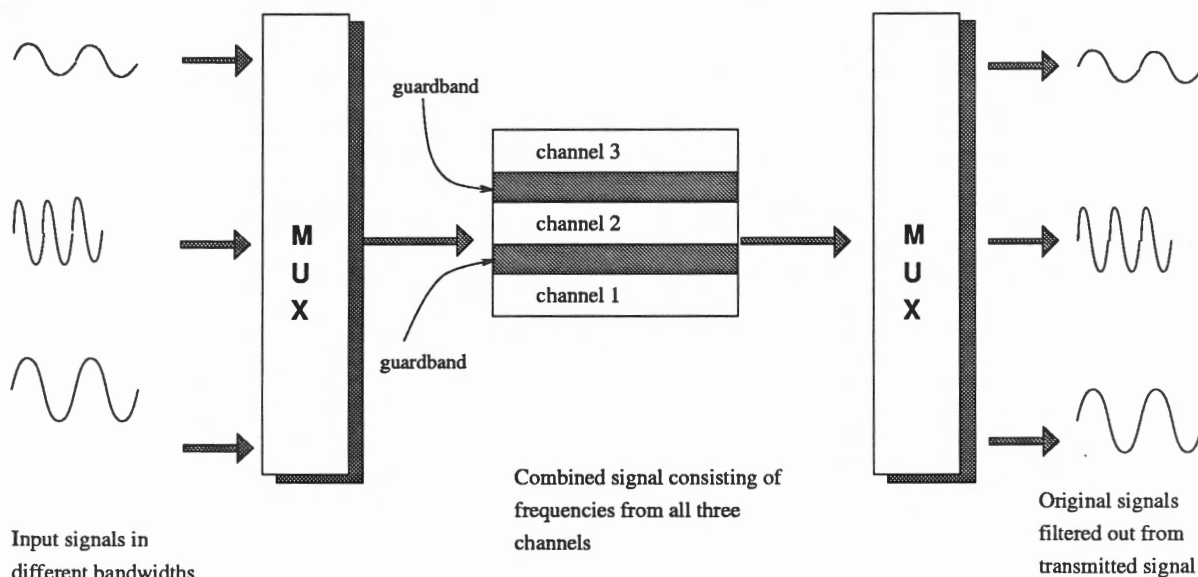


Figure 5.6: Wavelength Division Multiplexing

down the channel access to a fixed portion of the frequency spectrum. In the case of optical signals, the light produced must therefore remain focused closely on a single wavelength, so that different signals of different wavelengths do not interfere with each other, a phenomenon called *crosstalk*.

The main problem with WDM is that each wavelength of light travelling down the channel needs a separate repeater, adding to the cost of the system as the distance travelled is increased.

5.5.4 Combining WDM and STDM

The highest capacity system would be a hybrid of STDM and WDM, allowing for the greatest possible use of a single link. A number of channels could be multiplexed together using STDM, until the practical limit

Code division Multiplexing: used, for instance, in CDMA (code division multiple Access) for the planned UMTS (Universal Mobile Telecommunication System). Every sender has a pin code. Successor to GSM

of the multiplexing and demultiplexing equipment is reached. Then each of the resultant channels will be converted to a specific wavelength and multiplexed into an optical fibre say, using WDM.

The level of technology in the case of optical fibre is so far developed that it is possible to send hundreds of individual signals down a single fibre thus allowing practically unlimited capacity without considering STDM. This technology is known as *Dense Wave Division Multiplexing (DWDM)* and is bound to upset the traditional view in telecommunications that capacity is limited and costly.

5.6 Digital Subscriber Line (DSL)

A -
H - DSL
V -

The largest challenge in the deployment of high-speed digital network to every office and household, the most challenging part is the link between the subscriber and the network service provider: the digital subscriber line or what is colloquially known as the *last mile*. Historically these twisted-pair links were installed to carry voice-grade signals in a bandwidth from 0 - 4KHz. The wires are however capable of carrying signals up to 1Mhz and more.

ADSL is the best known of a family of new technologies designed to provide high-speed digital data transmission over ordinary telephone wires and is now an ANSI standard. The term *asymmetric* refers to the fact that that ADSL provides more capacity downstream from the service provider to the subscriber than upstream. The rationale being that most user transmissions are in short commands or e-mail messages, whereas incoming traffic such as HTML pages can involve large amounts of data and include images or even video.

ADSL uses frequency-division multiplexing (FDM) to exploit the 1 MHz capacity of a twisted pair in the way illustrated in Fig 5.8.

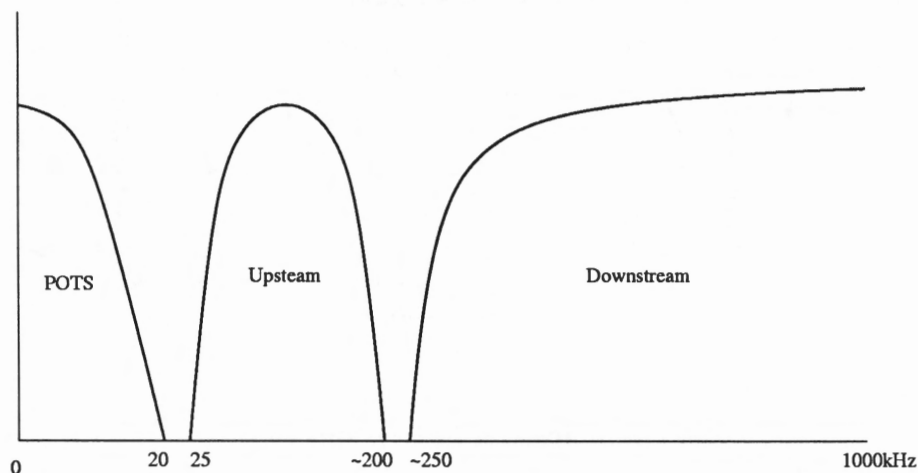


Figure 5.7: Asymmetric Digital Subscriber Line (ADSL) technology using FDM only

1. The lowest 25Khz is reserved for the voice channel, known as POTS (plain old telephone service). The voice channel requires only the 0 - 4 KHz band; the additional bandwidth is to prevent interference or *crosstalk* between the voice and data channels.
2. Use a technique known as echo-suppression or otherwise FDM to allocate two bands, a narrower upstream band and a wider downstream band. Echo-suppression is a signal processing technique that allows the transmission of digital signals in both directions simultaneously on a single transmission

line. Essentially, a transmitter must subtract the echo of its own transmission from the incoming signal to recover the signal sent by the other side as illustrated in Fig 5.8.

3. Use FDM within each of the upstream and downstream bands where a single bit stream is split into multiple parallel bit streams individually carried in a separate frequency band.

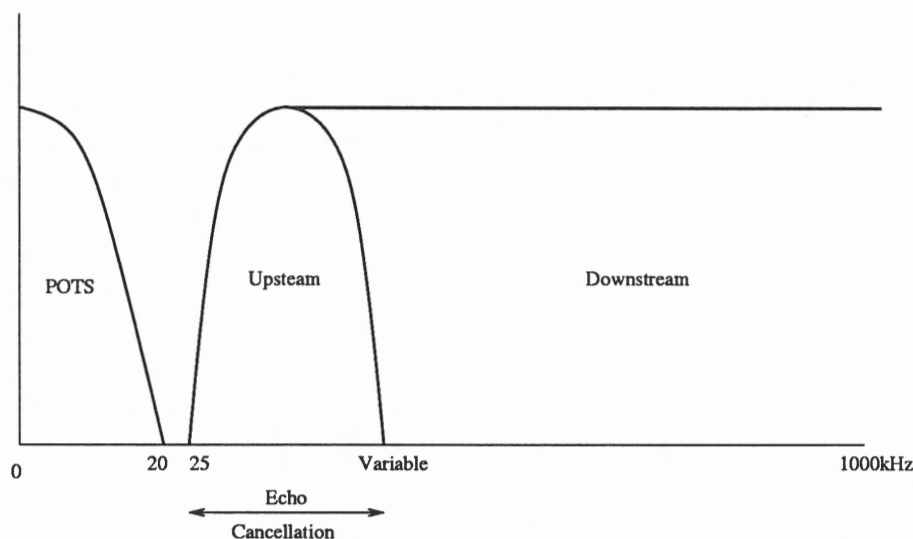


Figure 5.8: Asymmetric Digital Subscriber Line (ADSL) technology using Echo-suppression

When echo-suppression is used, the entire frequency range for the upstream channel overlaps the lower portion of the downstream channel. This has two advantages compared to the use of distinct frequency bands for upstream and downstream.

1. The higher the frequency, the greater the attenuation. With the use of echo-suppression, more of the downstream bandwidth is in the lower range.
2. The echo cancellation design is more flexible for changing the upstream capacity should this be required. The upstream channel can be extended upward without running into the downstream; the area of overlap is merely extended.

The disadvantage of the use of echo-suppression is the need for echo-suppression logic on both ends of the line. ADSL provides a range of up to 5 Km depending on the diameter of the twisted pair cable and its quality as well as the quality

ADSL is only one of a number of schemes recently proposed and collectively known as xDSL.

HDSL High Data Rate SDL was developed to provide 1.544Mbps (the American T1 rate, see Sec. 6.3 on page 75). It is a symmetric service which uses an encoding scheme known as 2B1Q to provide the required data rate over two twisted pair lines within a bandwidth that extends only up to 196KHz. The low frequency allows ranges of about 3.7Km to be reached easily.

SDSL Although HDSL is attractive for replacing existing T1 lines, it is not suitable for the traditional last mile since the typical residential subscriber has a single twisted pair to his house whereas HDSL requires two twisted pairs. SDSL was developed to provide the same type of service as HDSL but over a single twisted pair using 2B1Q encoding as well. Echo-suppression is used to achieve full duplex transmission over the single pair. Note that SDSL does not support POTS.

UDSL Universal SDL is the European equivalent of HDSL in that it will support E1 trunk rates of 2.048 MBps rather than the T1 rate of 1.544Mbps (see Sec. 6.3).

VDSL VDSL is an asymmetric DSL service for twisted pair lines that is in the process of being defined at the time of writing. Transmission rates are expected to range from 13 to 52Mbps downstream and upstream rates ranging anywhere from 1.5Mbps to an upstream rate equivalent. Maximum distances will range to 1.2Km. VDSL will most likely be deployed as the last mile part of a fibre-copper hybrid local loop in which fibre is used for the service provider-neighbourhood link and VDSL over UTP is used for the neighbourhood-residential home link. Services will include Internet access, video-on-demand and high-definition (digital) television (HDTV), amongst others.

5.7 Switching Techniques

Finally in this chapter we need to distinguish between the three most common ways of *switching* links between two communicating parties. All of the xDSL techniques discussed in the previous section provide a permanent point-to-point link between service provider and the user and is available whether being utilised or not.

5.7.1 Circuit Switching \equiv connection oriented

Similarly when you (or your computer) places a telephone call, the switching equipment within the telephone system seeks out a physical “copper” path all the way from your telephone to the receiver’s telephone. This technique is called *circuit switching* and is shown schematically in Fig. 5.9.

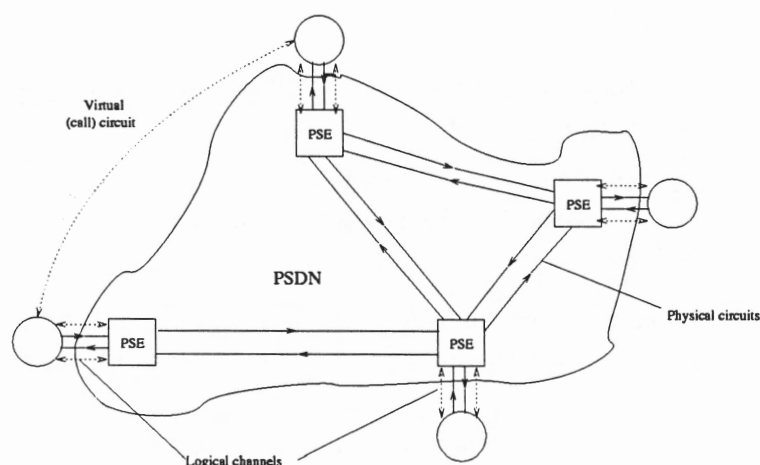


Figure 5.9: Circuit switching

Each of the four rectangles represents a carrier switching office (called an end office, toll office, etc). In this example, each office has three incoming lines and three outgoing lines. When a call passes through a switching office, a physical connection is (conceptually) established between the line on which the call came in and one of the output lines, as shown by the dotted lines. In the early days of telephony, the connection was made by having the operator plug a jumper cable into the input and output sockets.

The model shown in Fig. 5.9 is highly simplified of course, because parts of the “copper” path between the two telephones may, in fact, be microwave links onto which thousands of calls are multiplexed. Neverthe-

less, the basic idea is valid: once a call has been set up, a dedicated path between both ends exists, and will continue to exist until the call is finished.

An important property of circuit switching is the need to set up an end-to-end path before any data can be sent. The elapsed time between the end of dialing and the start of ringing can easily be 10 sec, more on long distance or international calls. During this time interval, the telephone system is hunting for a copper path, as shown in Fig. 5.9. Note that before data transmission can begin, the call request signal must propagate all the way to the destination and be acknowledged. For many computer applications, (e.g., point-of-sale credit verification), long setup times are undesirable.

As a consequence of the copper path between the calling parties, once the setup has been completed, the only delay for data is the propagation time for the electro magnetic signal, about 6 milliseconds per 1000Km. Also as a consequence of the established path, there is no danger of congestion i.e., once the call has been put through, you never get busy signals, although you might get one before the connection has been established due to lack of internal switching or trunk capacity.

5.7.2 Packet Switching

connections

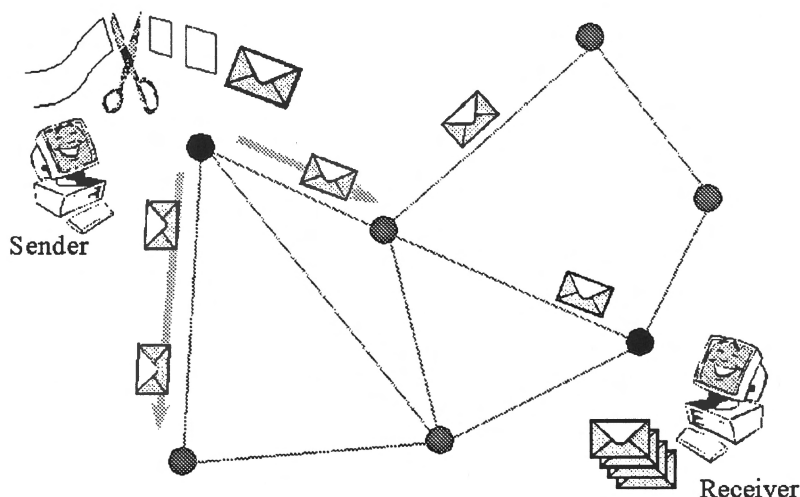


Figure 5.10: Conceptual view of packet switching

When the services provided over a channel occur in real time such as is the case with voice or video a dedicated circuit is necessary because of the low tolerance for delays. Computer communication sessions are different in that transmission occurs in spurts of a few milliseconds and delays are in general not that critical. An e-mail message or web pages delivered to a remote server can afford to wait a few seconds. This nature of computer communication was recognised very early on and a technique called *packet switching* was invented to allow many users to share the same communication channel. Rather than devote the entire circuit to a user each message sent between users is *packetised*, each packet given the address of the sender and receiver, much like a letter, and a sequence number and sent off individually to its destination. At the receiving end the packets, if received without error, is reassembled in sequence into the message either

by the user or the network depending on the service provided. Fig. reffig:packetswitching illustrates this principle.

Note that each packet need not travel along the same route from source to destination and because packets are numbered, individual packets can resend by the origin if required. Much like cars travelling in a city network, if the one route is congested or even closed, another route can be followed to get to the destination.

5.7.3 Cell Switching

*connection oriented -
circuit has to be set up.*

Cell switching is newer concept which is similar to packet switching. Whereas most protocols allow packets to be of any length, cells are of a fixed 53 octet³ length. Cells do not carry a source address however, as we shall learn in Chapter 10 and always follow the same route from source to destination.

with a 5 byte header

5.8 Exercises

Exercise 5.1 Explain what is meant by the term multiplexing. Explain the difference between time division and wave length division multiplexing?

Exercise 5.2 Why is frequency division multiplexing better suited to fibre optics than copper cables?

Exercise 5.3 A voice line needs to transmit one byte of information every 125 μ s. Assuming that the electrical-optical interface of a fibre optic system takes 625 ns to process one bit, how many voice lines can be multiplexed through the fibre using TDM?

Exercise 5.4 Can fibre be used to transmit analogue data? Explain.

Exercise 5.5 Twenty-four voice channels are to be multiplexed and transmitted over a twisted pair. What is the bandwidth required for FDM? Assuming a bandwidth efficiency (ratio of data rate to transmission bandwidth) of 1bps/Hz, what is the bandwidth required for Synchronous TDM using PCM?

Exercise 5.6 Ten 9600bps lines are to be multiplexed using TDM. Ignoring overhead bits, what is the total capacity required for synchronous TDM? Assuming we wish to limit the average line utilization to 80 percent and assuming each line is busy 50 percent of the time, what is the capacity required for statistical TDM?

Exercise 5.7 A character interleaved (multiplexed) time-division multiplexor is used to combine the data streams of a number of old-fashioned 110bps asynchronous terminals for data transmission over a 2400bps digital line. Each terminal sends asynchronous characters consisting of 7 data bits, 1 start bit and 2 stop bits. Assume that one synchronous character is sent every 19 data characters. Determine the number of

1. bits per character;
2. number of terminals that can be accommodated by the multiplexor.

³An octet, as the name implies, is 8 bits, or in other words a byte long. The former term is more frequently used in the telecommunications industry for reasons which are not clear.

Chapter 6

Physical Data Communication Interfaces, Standards and Errors

6.1 Objectives of this Chapter

All signals which carry the logic of peer-to-peer communication travel along physical channels. In this chapter we cover the most general protocols required at the Physical layer.

1. The electrical interfaces we need to know about for networking are RSC232 and X.21. There are many others, but they are not as common as those we discuss.
2. In the optic signal world the standards are mercifully fewer, but still there are SONET and SDH which are not the same, but are used for the same thing: Optic network interfaces.
3. Transmission errors are a fact of life, however good the transmission media are. The mere fact that a medium is not perfect requires one to check for errors in data communication. On completing the section on errors, the reader should know about Hamming distance and the difference between error correction and detection, including ordinary and block parity.
4. The most important technique for error detection covered in this chapter is Cyclic Redundancy Coding which, in its variations, is used almost exclusively for that purpose.

While some may consider the physical links and their standards the domain of the electrical engineers and technicians, the computer scientist who does not understand what is happening at that level, will certainly not be as good in understanding the protocols and technologies at the higher layers as the one who does not have this knowledge. That is why this chapter is included.

	Analog	Digital
Parallel	?	Parallel Cable-bus
Serial	RS232-c	X.21

This section focuses on the standards which apply at the *interface* between the user equipment (also called the *Data Terminal Equipment (DTE)* in the ITU-T standards) on a network and the network itself or *Data Circuit-terminating Equipment (DCE)*. An interface can be defined as the line of demarcation between two pieces of equipment which, in order to operate harmoniously, must each obey a complementary set of interface specifications.



6.2 Electrical Signal Interfaces

Both serial interfaces where a single bit is transmitted per bit period and parallel interfaces, where all the bits constituting a character are transmitted during one bit period in parallel across the interface, are used in practice. The latter interface is normally associated with high speed devices, such as a printer, or when the terminal is connected to a bus, such as the IEEE 402 standard, and in all cases carry digital signals. In this section we shall concern ourselves with the more common serial interfaces.

6.2.1 Serial Analog Interface

One well-known standard of physical inter-connection is the *RS232C Serial Interface* which, functionally is the same as the international ITU-T Standard V24. This standard was developed by Electronics Industries Association (EIA) and covers interfaces for speeds up to 20 000 bps. At higher speeds, ITU-T specification V35 applies.¹

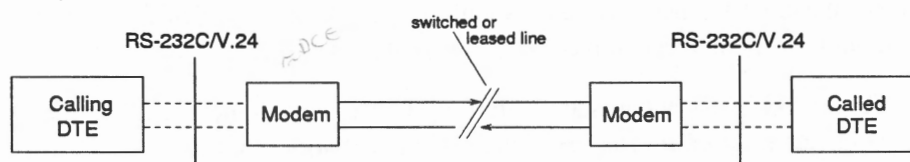


Figure 6.1: RS-232C/V.24 interface

V24 specifies the hardware and electrical interface between a *synchronous terminal*, i.e., a DTE using a synchronous protocol (see Sec 5.4) and the modem on the one hand and the modem and a host computer on the other hand. An equivalent ITU-T recommendation V21 applies to the interface between an *asynchronous DTE* and a modem. In fact, the standard also applies to a remote DTE connected via an analogue circuit to a DTE, but our description will be in terms of the former configuration. It is important to remember that in either case the two pieces of digital equipment being connected are remote and that the connecting lines carry analog signals, in other words, modems are involved.

Since manufacturers like to supply terminals which can be coupled to a computer either via a modem or directly (when the distance involved is no more than about 50 meters) RS232C also caters for this latter situation. Since the signals passing between a terminal and host computer are all digital it makes more sense however, to send digital signals straight down the line via a simple interface commonly referred to as a *current loop interface*.

The V24 recommendation includes a large number of circuits identified by function, name and number, each of which represents a pin (see Figure 6.2) on the 25-pin standard connector. However, the allocation of circuit to pin is in fact part of an ISO (International Standards Organization) specification, ISO 2110, rather

¹ All ITU-T V-series recommendations are for data transmission using the telephone network.

than ITU-T. Many of the circuits are rarely used, and some are applicable only to synchronous modems or modems with more elaborate facilities such as secondary backward channels. The discussion of every connection is a technical issue which can be looked up in any manual, but we shall describe the role that some of the circuits play in a typical DTE-DCE connection. Suppose the DTE is a PC and the DCE is a

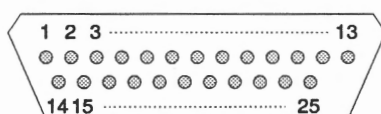


Figure 6.2: RS-232C Connector

modem. The first 6 circuits are primarily used to establish an exchange that ensures that neither device will send data when the other is not expecting it. Fig. 6.3 shows the exchange over a period of time. Since the

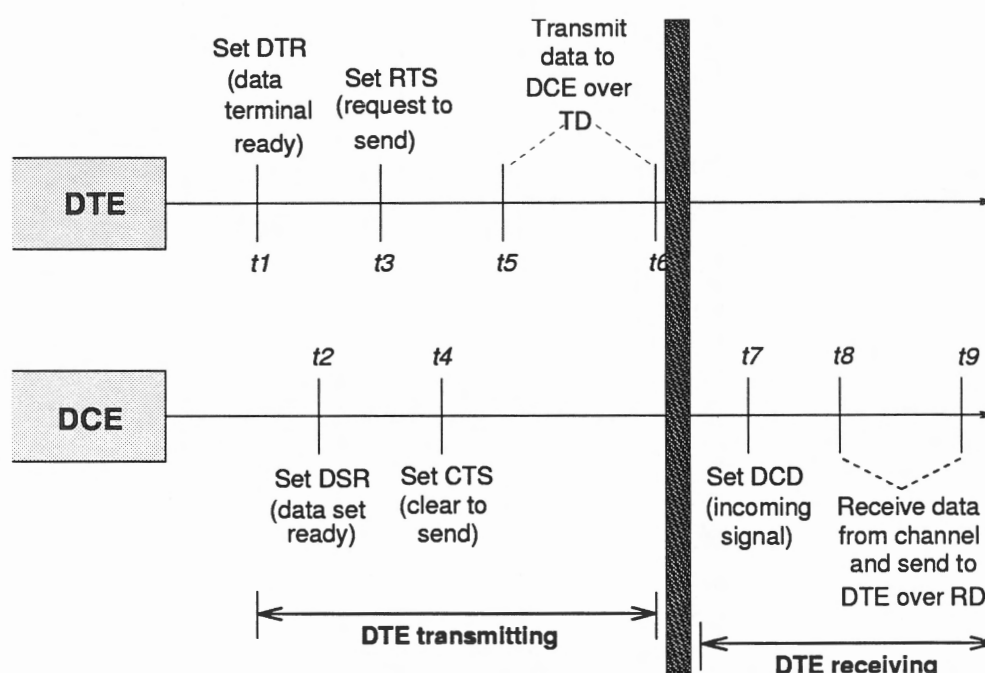


Figure 6.3: Sending and receiving over an RS-232 connection

DCE interfaces with the network on behalf of the DTE, it must know when the DTE is ready. The DTE does this by asserting (sending a signal) DTR circuit number 20 (time $t1$ in Fig. 6.3). The DCE senses the signal and responds by connecting (if it has not already done so) to the network. Once the DCE has connected and is also ready, it asserts DSR circuit number 6 (time $t2$). Effectively, the DCE acknowledges the DTE's state of readiness and declares it also ready.

Once they are both ready, the DTE requests permission to transmit data to the DCE by asserting RTS circuit 4 (time $t3$). This circuit also controls the direction of flow in the half-duplex communications. On sensing RTS, the DCE enters "transmit mode", meaning it is ready to transmit data over the network. It responds by asserting the CTS circuit number 5 (time $t4$). Finally, the DTE sends data over TD circuit number 2 (between times $t5$ and $t6$).

When the DCE detects an incoming signal from the network it recognises, it asserts DCD circuit number 8 (time $t7$). As the signals come in, the DCE sends them to the DTE using RD circuit number 3.

6.2.2 Serial Digital Interface

As early as 1969, ITU-T realized that eventually carriers would bring true digital lines onto customer premises. To encourage compatibility in their use, ITU-T proposed a digital signaling interface, X21² which was approved in 1976. The standard specifies how the customer's computer, or DTE in ITU-T nomenclature, sets up and clears calls by exchanging signals with the carrier's equipment, or DCE for *synchronous communication*. The equivalent standard for *asynchronous communication* is X20. X20 and X21, like V24/RS232 are at the *physical level* of the OSI reference model.

Before embarking on a description of the X21 interface it is worth while to remind ourselves what we learned in Sec. 1.6 on page 11 out that communication facilities perform in four time phases:

1. The *call-establishment phase* concerns the "setup" of the communications carrier facilities so as to provide communication between a calling DTE and a called station.
2. The *data transfer phase*, when the connection is established and data may be transferred.
3. The *call-clearing phase* concerns the release of the communications carrier facilities.
4. The *idle phase* equates to the on-hook condition in telephone networks, but may further indicate that the DTE is Ready or Not Ready to accept a call.

Note that not all four phases need be present. The V24/RS232C interface described previously only allows for phases 2 and 3 since the terminal is permanently connected to the host computer. The names and

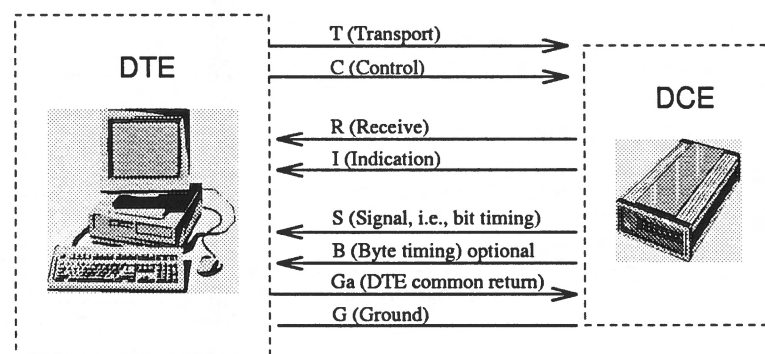


Figure 6.4: Signal lines used in the X21 serial digital interface

functions of the eight wires defined by X21 are given in Fig. 6.4. The physical connector has 15 pins, but not all of them are used. One of the basic principles agreed to early in the development of X21 was that the interface should be transparent, that is, *bit-sequence independent* for all data in the data transfer phase. Simplicity was also a prime objective as evidenced by the resulting circuits shown.

The DTE uses the T and C lines to transmit data and control information, respectively. The DCE in turn uses the R and I lines for data and control, thus allowing for full duplex communication. Since synchronous operation is specified, a third line provides bit timing from the network to the DTE. At the carrier's option, a B line may also be provided to group the bits into 8-bit frames. If this option is provided, the DTE must begin each character on a frame boundary. If the option is not provided, both DTE and DCE must begin every control sequence with at least two special synchronisation SYN characters, to enable the other one to deduce the implied frame boundaries. In fact, even if the byte timing is provided, the DTE must send

² All X-series recommendations of ITU-T refer to data transmission using new data communication networks.

the two SYNs before control sequences, to maintain compatibility with networks that do not provide byte timing.

The alphabet used to exchange control information is that of the International Alphabet No 5 (ITU-T Recommendation V3). These control characters are sent on the T and R circuits preceded by two SYN characters (when not in the data transfer phase). This use of *code strings* (rather than circuits) for control information is a major advantage of X21 in that it provides a practically unlimited number of control signals. ON and OFF conditions for circuits C and I refer to continuous ON (binary 0) and continuous OFF (binary 1) conditions.

Although X21 is a long and complicated document, which references other long and complicated documents, the *state transition* diagram in Fig. 6.5 illustrates the main features. In that diagram we shall show how the DTE places a call to a remote DTE, and how the originating DTE clears the call when it is finished. Bear in mind that X.21 defines the interface between the DTE or user equipment and the network, or DCE and the protocols at the *calling* and *called* end are identical. When referring to C and I, we will follow ITU-T practice and call 1 OFF and 0 ON. When the line is idle (i.e., no call present), the four signaling lines are all *one* as illustrated in state 1 in Fig. 6.5. When the user/DTE wants to place a call, it sets T to 0 and C to ON. When the network/DCE, in response to this, is ready to accept a call, it begins transmitting the ASCII “+” character on the R line, in effect, a digital dial tone, telling the user/DTE that it may commence dialing or, in other words, sending the selection signals.

The user/DTE “dials” the number by sending the called user/DTE’s address as a series of ASCII characters using the T line, one bit at a time shown as state 4.

Now a series of exchanges take place between the calling DCE and the called DCE reflected in states 6A, 6B, 6C and 7 and 10. These *call progress signals* arrive to the DCE/network from the called DCE to inform the calling DCE of the called DCE status. The *call progress signals*, defined in ITU-T recommendation X96, consist of two digit numbers, the first of which gives the general class of the result, and the second the details. The general class include: *call put through, try again* (e.g., number busy), *call failed and will probably fail again next time* (e.g. access barred, remote DTE out of order, DTEs incompatible), *short term network congestion*, and *long term network congestion*. If the call can be put through, the DCE sets I to ON to indicate that the data transfer may begin. At this point a full-duplex digital connection has been established (state 13 in the diagram), and either side can send information at will.

Either DTE can end the connection by setting its C line to OFF. Having done so, it may not send more data, although it must be prepared to continue receiving data until the other DTE has finished.

At the called side, the protocol machine changes state from state 1 to state 8, 9 and 10bis³.

In the event that a call collision occurs, i.e., an incoming call and an outgoing call take place simultaneously call is canceled and the outgoing call is put through. ITU-T made this decision because it may be too late at this point for some DTEs to reallocate resources already committed to the outgoing call.

Carriers are likely to offer a variety of special features on X21 networks such as *fast connect*, in which setting the C line ON is interpreted by the DCE as a request to re-connect to the number previously dialed. This feature eliminates the dialing stage, and might be useful, for example, to place a separate call to a time-sharing computer every time the person at the terminal hits return. Another possible X21 option is the *closed user group*, by which a group of customers (e.g., company offices) could be prevented from calling or receiving calls from anyone outside the group. Call redirection, collect calls, incoming or outgoing calls barred and caller identification are other possibilities.

As an interim measure until all devices are digital (which is not soon), the connection to public data networks of synchronous DTEs, which are designed for interfacing to “old” synchronous V-series (RS232C in USA)

³*bis* is a French abbreviation for “the second” — state 10 in other words in this case

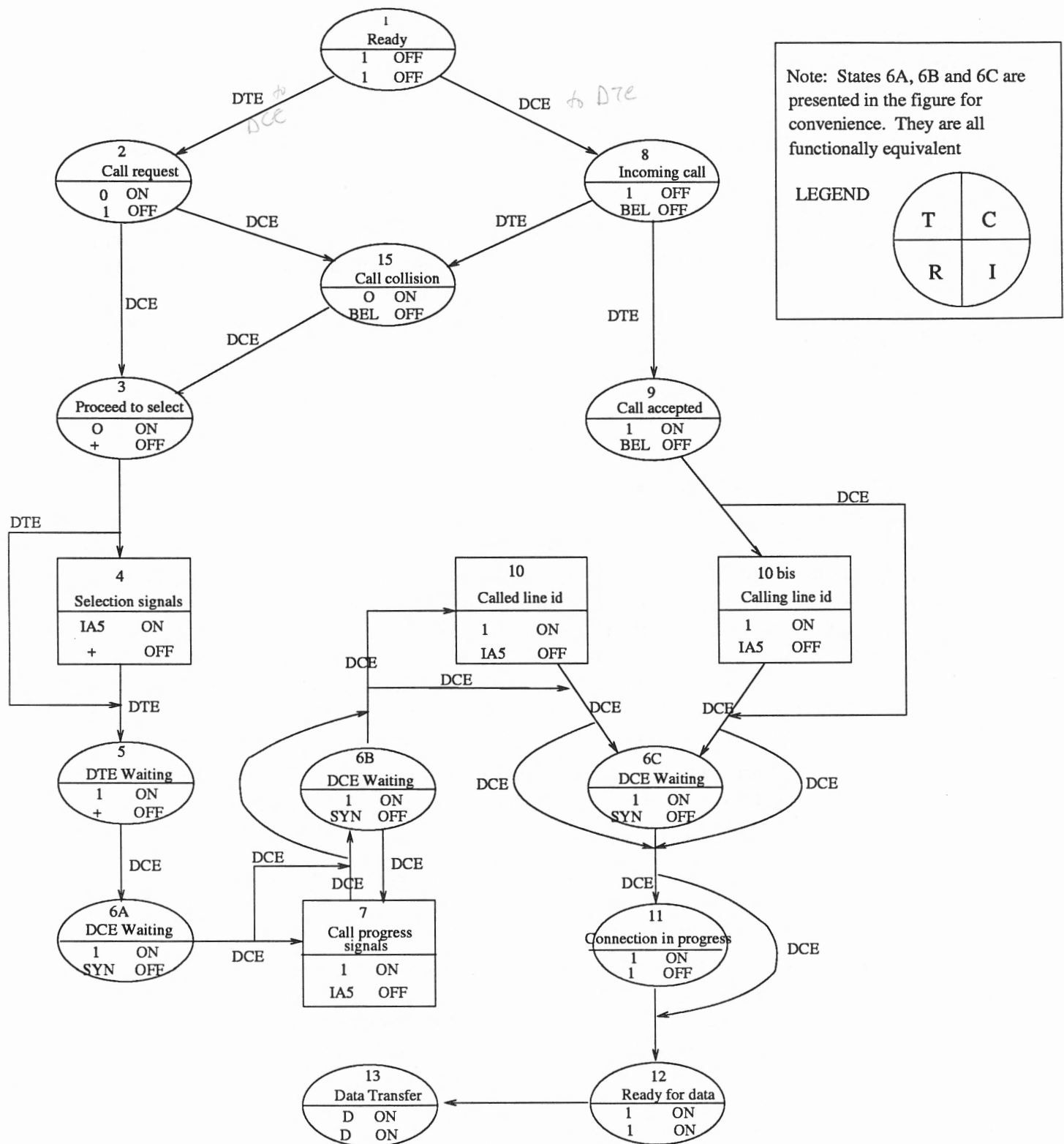


Figure 6.5: X21 protocol state transition graph

modems. This conversion interface facility is called *X21 BIS* (*X20 BIS* for asynchronous communication). It seems that there is no change required to existing products (that now use the V24 or RS232C interface) to attach to an X21 network when using X21 BIS. The user may, however, pay extra for the X21 BIS interface, and many new facilities offered by X21 (for example, call-progress commands), are then not available to the user.

6.3 Optic Signal Interfaces

Two optic signal interface standards, one from the United States and, for once agreed by Japan and one from Europe, define a means of transmitting data at rates up to 2488.32 Mbps. These are the SONET (Synchronous Optical Network) and SDH (Synchronous Digital Hierarchy), protocols for broadband digital services. In this context, broadband refers to "broad bandwidth", in other words, channels that carry a high capacity.⁴ Strictly speaking, *broadband services* are those requiring a transfer rate greater than 1.544 Mbps in North America (designated the *T1 rate*) and 2.048 Mbps in Europe (called the *E1 rate*), these rates are also known as the *primary transfer rates*. The

8 bits

56 bits

64 bits

Type	Digital Bit Rate	Voice Circuits	T-
DS-1	1.544 Mbps	24	1
CEPT1	2.048 Mbps	30	-
DS-1C	3.154 Mbps	48	2
DS-2	6.312 Mbps	96	4
DS-3	44.736 Mbps	672	28

7 bits

Table 6.1: Typical payloads.

international standard for digital transmission is SDH, adopted by the ITU-T in its recommendations G.707, G.708 and G.709. SDH incorporates SONET principles. Services provided by SONET and SDH range from 64 kbps (the DS-0 signal which carries one telephone channel) to 2488.32 Mbps. Table 6.1 lists payloads supported by SONET/SDH.

In the SONET system, three types of equipment are used, as shown in Fig. 6.6. *Path-terminating equipment* signifies the end of the transmission line. The path is an end-to-end transmission line that connects two SONET terminals or switches, and includes both electrical and optic components. *Line-terminating equipment* connects SONET hubs by means of optic fibre. The SONET hubs multiplex several transmission lines from SONET terminals onto a single optic transmission line. *Section-terminating equipment* comprises the repeaters and amplifiers in the optic cable. Amplifiers increase the signal power without halting the transmission of a signal; repeaters receive, amplify and retransmit signals to counter the effects of signal loss.

SONET takes digital channels of various capacities and maps them onto a basic transmission rate unit called the Synchronous Transport Signal - Level 1 (STS-1). The STS-1 transmits frames of 810 bytes or 6480 bits, at a rate of 8000 frames/second. A frame thus takes 125 μ s to transmit, the chief reason being that a telephone call requires the transmission of one octet every 125 μ s. STS-1 thus has a basic transfer rate of 51.840 Mbps (6480 * 8000 = 51 840 000). Higher level signals are obtained by using add-drop multiplexers

⁴This usage of the word "broadband" is not to be confused with broadband coaxial cable, which can carry multiple analog channels simultaneously.

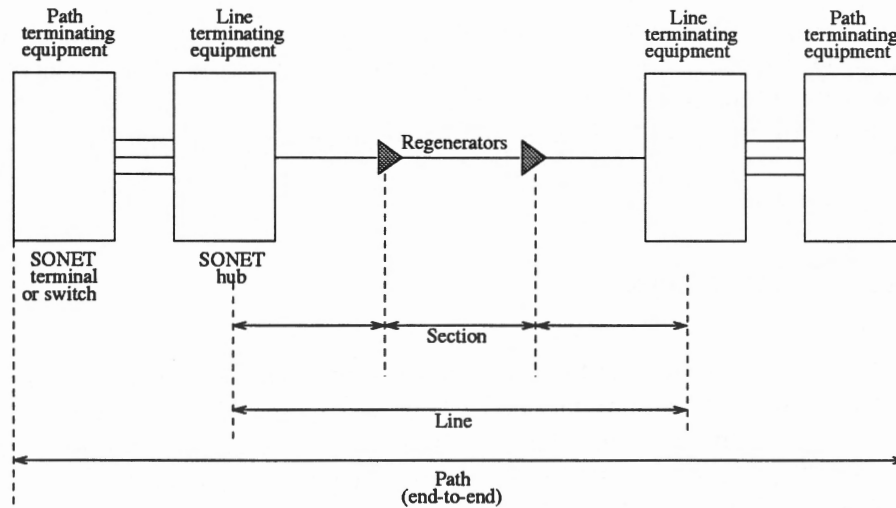


Figure 6.6: SONET configuration.

(ADMs) to multiplex (by byte interleaving the incoming signals) N STS-1 channels together to form an STS- N channel, with N times the capacity of the STS-1 channel. For example, the STS-3 channel transmits at a rate of 155.52 Mbps.

The frames transmitted on the STS-1 channel are 90 by 9 octets (8-bit bytes) (see Fig. 6.7). The first three columns of the frame contain transport overhead (27 bytes), with 9 bytes allocated for section overhead and 18 bytes allocated for line overhead. The remaining 87 columns comprise the STS-1 envelope capacity, including another 9 bytes which are reserved for the path overhead. The effective user capacity is thus 86 columns by 9 octets, giving 774 bytes and an effective transmission rate of 49.536 Mbps. Section overhead controls data integrity between the regenerators, line overhead ensures correct transmission of the STS- N signals by the ADM multiplexers, and path overheads carried end-to-end, only being checked at the final destination.

SONET takes incoming digital channels of any transmission speed and sends them down the transmission line. Fig. 6.8 shows an example of the schemes used to multiplex the various signal speeds onto the basic 51.84 Mbps transfer rate. Service adapters map the incoming signals into STS-1 envelopes. If the transmission rate of the incoming line is below the STS-1 rate, then it is converted into Virtual Tributaries (VTs); several of these VTs are multiplexed into one STS-1 channel. Examples of sub-STS-1 channels are DS-0 (64 kbps) and DS-1 (1.544 Mbps). DS-3, with a speed of 44.736 Mbps, is converted straight into an STS-1 signal. Higher speed signals are initially converted into M STS-1 signals (with $M \leq N$, where N is the final channel speed) before being multiplexed into STS- M or STS- Mc signals. STS- Mc stands for Concatenated STS- M ; each STS- Mc envelope carries just one copy of the path overhead, whereas an STS- M signal carries M copies of the path overhead and is thus wasteful of the bandwidth. An STS- Mc signal can only be sent if there are no intervening multiplexers between the sender and the receiver, it sets up a high speed single channel between the two multiplexers for signals requiring high bandwidth. Once the service adapters have created the STS-1 and STS- Mc channels, they are byte-interleaved by the add-drop multiplexers to form STS- N signals, and sent to the electrical/optic converter. At this point, the electrical signals are scrambled to ensure that there are enough signal changes for synchronisation to work, and then converted into an OC- N (Optical Carrier - Level 1) optic signal. The OC- N signal which traverses the optic fibre, carries the same transmission rate as the STS- N signal.

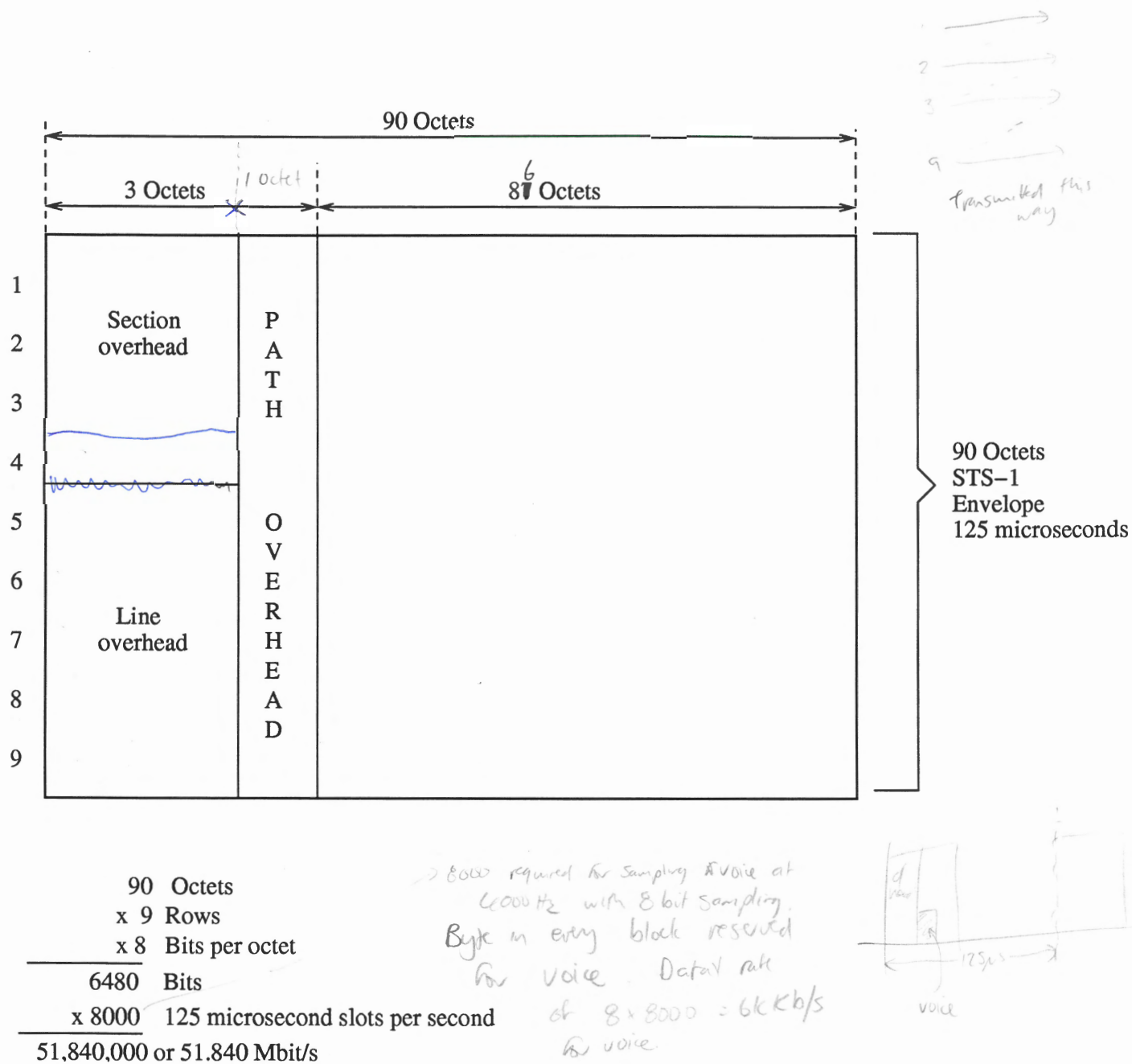


Figure 6.7: The STS-1 SONET frame.

Write out actual data which can be transmitted.
Compare to rate shown in Table 6.1

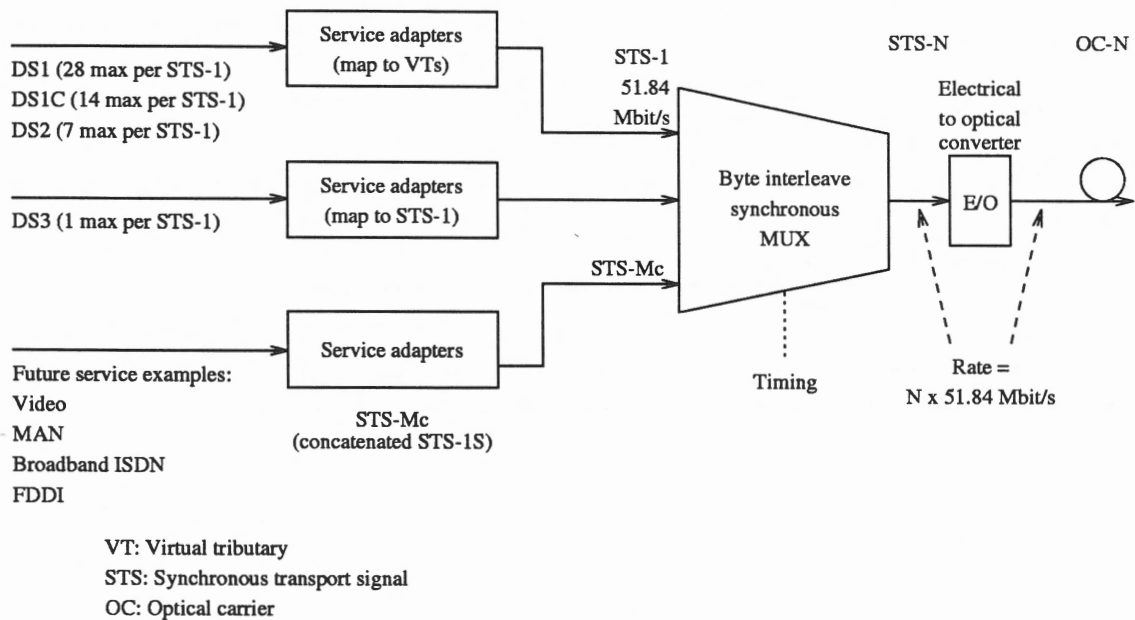


Figure 6.8: Mapping basic signals onto an STS-N channel.

6.3.1 Synchronisation and SONET/SDH

Synchronisation is a tricky issue to define exactly. Strictly speaking, to qualify as *synchronous*, the source and destination must be clocked by the same timing reference; signals that are sent and received will always have exactly the same frequency, and cannot fall out of alignment. However, this definition means that SONET cannot be synchronous, since the sender and receiver may be situated very far from each other. In the SONET system sender and receiver send signals arbitrarily close to a predetermined frequency (e.g. STS-1); the signal is referred to as *plesiochronous*. Over time, the sender and receiver will become skewed from each other, and frames would need to be repeated or deleted to handle buffer overflow and underflow. SONET allocates three bytes of line overhead to pointers to solve this problem. When the frames become out of alignment, these pointers are used to realign the frames. For example, if a DS-3 frame is imbedded in an SDH frame, the pointer points to the start of the DS-3 frame. When the frame is received, the receiving station can then point directly to the DS-3 frame. If this frame is out of alignment, the pointer is just shifted up or down until it is again synchronously aligned.

6.3.2 The relationship between SONET and SDH

SONET is the optic protocol adopted by the North Americans and Japanese, whereas SDH is a worldwide protocol standard adopted by the ITU-T that incorporates SONET, and is used mainly in Europe. SONET's basic transmission rate is the STS-1 at 51.84 Mbps, whereas SDH operates at the STM-1 (Synchronous Transport Mode - Level 1) rate of 155.52 Mbps. Table 6.2 lists the relationship between STS and STM signal speeds. The DS signals are of North American origin, while CEPT1 falls into the European family of signals. Although it would seem that STM-1 is equivalent to STS-3, it is actually only equivalent to STS-3c. STS-3c is the signal sent between two multiplexers with no other multiplexers in between the sender and the receiver, it is the concatenation of 3 STS-1 signals with only one copy of the path overhead present for each STS-3c signal. An STS-3 signal, by contrast, contains three separate copies of the path overhead. Further minor differences between SDH and SONET exist in the overhead bytes, which have different positions and

optical circuit

OC-level	SONET	SDH	Line Rate
OC-1	STS-1	-	51.84 Mbps
OC-3	STS-3	STM-1	155.52 Mbps
OC-12	STS-12	STM-4	622.08 Mbps
OC-24	STS-24	STM-8	1244.16 Mbps
OC-48	STS-48	STM-16	2488.32 Mbps
OC-192	STS-192	STM-64	9953.28 Mbps

Table 6.2: Transmission speeds of SONET and SDH.

functions in the two protocols, but do not present too many problems for inter-operability. Current physical interface chips that are produced to support SONET usually also include an STM operation mode, thus allowing switches using these chips to support either STS-3 or STM-1.

6.4 Transmission Errors

Physical transmission errors are a fact of life. As with the links in a chain, a communication channel is only as good as its weakest link. On microwave links and particularly fibre-optic channels (with an error rate of about 10^{-12}), errors occur very infrequently. Some links will still use a telephone line somewhere though, and with the 40 year depreciation time on existing telephone plant, transmission errors are going to continue being a fact of life. In this section, we shall investigate some of the problems more closely and see what can be done to overcome them.

6.4.1 Sources of Transmission Errors

Transmission errors are caused by a variety of different phenomena as we learned in Sec. 2.4.3 on page 30. Apart from noise we are reminded that the following are also sources of error:

- *Crosstalk* can occur between two wires which are physically adjacent.
- Microwave links are subject to fading, off course birds (getting partially roasted in flight) and the like.
- For voice transmission, it is desirable to compress the signal amplitude into a narrow range, because the amplifiers are not linear over wide ranges. This compression, called *compounding*, can introduce errors.
- It is not possible to produce a perfect carrier wave. Its amplitude, frequency and phase will always exhibit some jitter.
- Errors are introduced whenever the receiver gets out of synchronization with the transmitter. Typically, it takes a few tenths of milliseconds to get back into synchronization, with all the data transmitted in the meanwhile delivered to the wrong destination.

6.4.2 Error Rates

The quality of a communication channel is measured by the *Bit Error Rate* which we shall designate by e . For a copper channel e is typically 10^{-8} while for optic fibre it is more likely to be about 10^{-12} , as mentioned.

If a block of data contains n bits, then in the absence of error clustering, the probability that the block is transmitted without errors is

$$(1 - e)^n.$$

Inversely, the probability that the block is in error is

$$1 - (1 - e)^n.$$

If $e \ll 1$, this can be expanded using a binomial expansion to give an approximate probability of block error equal to en . The empirical data, however, gives a block error probability roughly equal to $10^{-4}n^{0.8}$ for blocks containing n 8-bit bytes and no start or stop bits. The results vary with line length, transmission speed, and other factors, so this is just an order of magnitude estimate.

In practice, however, as a result of the physical processes causing the noise, errors tend to come in bursts rather than singly. Having the errors come in bursts has both advantages and disadvantages over isolated single-bit errors. On the advantage side, computer data are always sent in blocks of bits. Suppose that the block size is 1000 bits, and the error rate is 0.001 per bit. If errors were independent, most blocks would contain an error. If the errors came in bursts of 100 however, only one or two blocks in 100 would be affected, on the average. The disadvantage of burst errors is that they are much harder to detect and correct than are isolated errors.

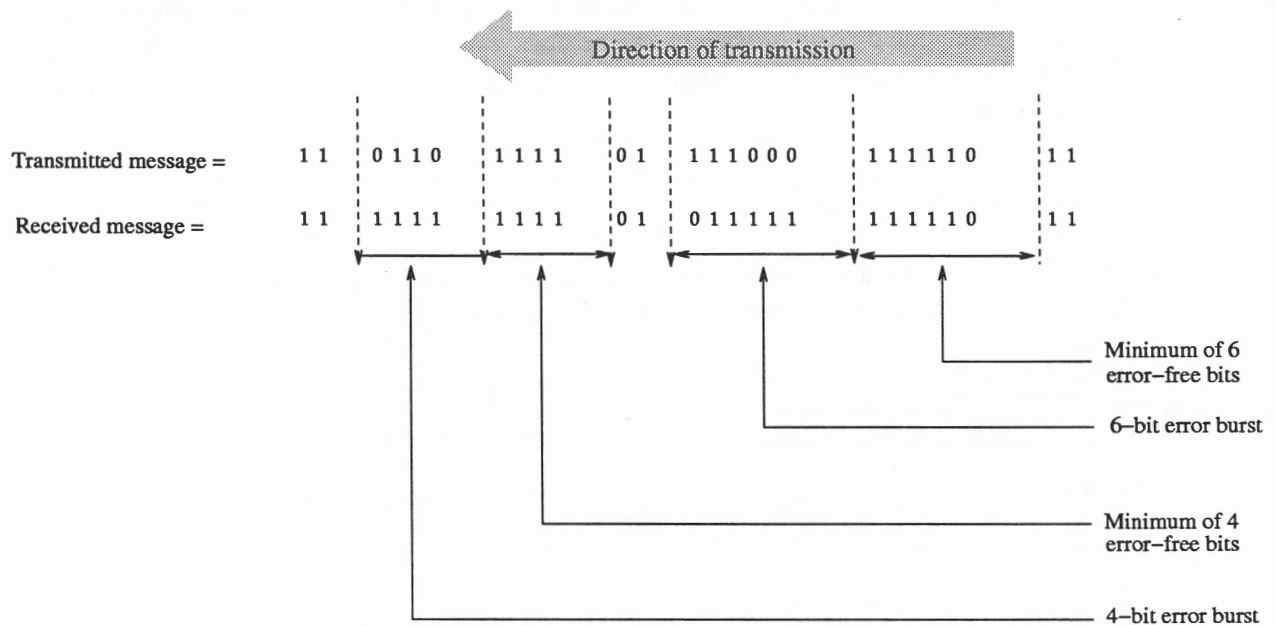


Figure 6.9: Error burst examples.

6.4.3 Error Codes

Network designers have developed two basic strategies for dealing with errors. One way is to include enough redundant information in each block of data sent to enable the receiver to deduce what the transmitted character must have been. The other way is only to include enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission. The former strategy uses *error-correcting codes* and the latter uses *error-detecting codes*.

Error correction - FEC (Forward error correction) we are not going to see it again
 Error detection, then correction.

6.4.4 Hamming Distance

Normally, a message consists of m message bits i.e., useful data), and r redundant, or *check bits*. Let the total length of the message be n bits, i.e., $n = m + r$. An n -bit unit containing data and check bits is often referred to as an n bit *codeword*.

Given any two codewords, say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just EXCLUSIVE OR the two codewords, and count the number of 1 bits in the result. The number of bit positions in which two codewords differ is called the *Hamming distance*.

Its significance is that if two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one into the other. When $(d - 1)$ single bit errors occur in such a code, no legal codeword can be converted into another legal code word and the error will be detected.

In most data transmission applications, all 2^m possible data messages are legal, but due to the way the check bits are computed, not all of the 2^n possible codewords are used. Given the algorithm for computing the check bits, it is possible to construct a complete list of the legal codewords, and from this list find the two codewords whose Hamming distance is minimum. This distance is the Hamming distance of the *complete code*. *The Hamming distance of a code is the minimum of the Hamming distance between any two codewords.*

The error-detecting and error-correcting properties of a code depend on its Hamming distance. As explained, to *detect* d errors, you need a distance $(d + 1)$ code. To *correct* d errors, one needs a distance $(2d + 1)$ code because that way the legal codewords are so far apart that even with d changes, the resultant pattern still differs in fewer bit positions from the original *legal* codeword than from any other *legal* codeword, and it can be determined uniquely. *To detect n bit errors, you need codewords with a Hamming distance of $n+1$. To correct n errors need distance $2n+1$.*

6.4.5 Parity

The most common method used for detecting bit errors with asynchronous and character-oriented synchronous transmission is to use *parity bits*. With this scheme, the transmitter adds an additional bit, the parity bit, to each transmitted character prior to transmission. The parity bit used is a function of the bits that make up the character being transmitted. Hence, on receipt of each character, the receiver can perform a similar function on the received character and compare the result with the received parity bit. If they are equal, no error is assumed, but if they are different, then a transmission error is assumed to have occurred.

Example 6.1 As a simple example of an error-correcting code, consider a code in which a single parity bit has been appended to the 9 bits of data. The parity bit of a codeword is computed by adding the number of 1 bits (modulo 2) and the parity bit is then chosen so that the total number of 1 bits, including the parity bit itself is either even (even parity) or odd (odd parity). Also, assume there are only four valid codewords: If the

0000000000 0000011111 1111100000 1111111111

codeword 0000000111 arrives with 2 bit errors, the receiver knows that the original could only have been 0000011111 (it assumes only two bits may have changed). If, however, a triple error changes 0000000000 into 0000000111, an attempt at error-correction will not work. This is because the code, with Hamming distance 5, can correct double errors only.

Let us consider the general case: Imagine that we want to design a code with m message bits and r check bits that will allow all single errors to be corrected. Each of the 2^m legal messages has n illegal codewords

Suppose we have an n -bit codeword: Invert every bit, in turn, in the codeword, to form a "family" of $(n+1)$, n -bit codewords eg codeword 1011 - 0
 make these into 1010
 1001
 1111
 0011
 if we receive this code, we know which family of codewords it comes from
 see to do next page

at a distance 1 from it. These are formed by systematically inverting each of the n bits in the n -bit codeword formed from it. Thus, to correct single bit errors, each of the 2^m legal messages requires $n + 1$ bit patterns dedicated to it. Since the total number of possible bit patterns is 2^n we must have

$$(n + 1)2^m \leq 2^n$$

Using $n = m + r$ this requirement becomes

$$(m + r + 1) \leq 2^r$$

Given m , this puts a lower limit on the number of check bits needed to correct single errors.

Even parity: XOR all bits in codeword to get 0
Odd parity

6.4.6 Error-Correcting Codes

This theoretical lower limit can, in fact, be achieved using a coding method due to Hamming. The *Hamming coding method* uses check bits placed among the data bits; each check bit covers a number of data bits. The check bits are placed in the “power of 2” positions, i.e., in bit position 1, 2, 4, 8, etc. *The check bits force EVEN parity on the bits it covers including itself.* On arrival at the receiver the parity is checked to determine whether an error occurred and the bit in error determined as we shall learn. Note the Hamming code only *corrects single bit errors*.

With 4 data bits 3 check bits are used as indicated in Table 6.3. The data bits are checked by the check bits

BIT POSITION	11	10	9	8	7	6	5	4	3	2	1
CHECK BIT				C_4				C_3		C_2	C_1
DATA BIT	D_7	D_6	D_5		D_4	D_3	D_2		D_1		
EXAMPLE	1	0	1	0	0	1	1	0	0	0	1

Table 6.3: The Hamming code explained

whose positions add up to the position of the data bit. For example, D_1 is covered by C_1 and C_2 because C_1 and C_2 are in positions 1 and 2 which add up to 3, the position of D_1 . Similarly, D_2 in position 5 is covered by C_3 and C_1 in positions 4 and 1 respectively. This is made clear further in Table 6.4.

DATA BIT	D_1	IN POSITION	3	IS COVERED BY	C_1	C_2	
DATA BIT	D_2	IN POSITION	5	IS COVERED BY	C_1	C_3	
DATA BIT	D_3	IN POSITION	6	IS COVERED BY	C_2	C_3	
DATA BIT	D_4	IN POSITION	7	IS COVERED BY	C_1	C_2	C_3
ETC							

Table 6.4: The Hamming code bit positions

The value of the check bits is calculated by forming the EXCLUSIVE OR (for EVEN parity) of the value of the data bits they cover as shown in the following example.

Example 6.2 Consider a codeword with the data bits 1010110:

	C_4	C_3	C_2	C_1
CHECK BIT POSITION	8	4	2	1
DATA BIT POSITION 3	-	-	0	0
DATA BIT POSITION 5	-	1	-	1
DATA BIT POSITION 6	-	1	1	-
DATA BIT POSITION 7	-	0	0	0
DATA BIT POSITION 9	1	-	-	1
DATA BIT POSITION 10	0	-	0	-
DATA BIT POSITION 11	1	-	1	1
EXCLUSIVE OR	0	0	0	1

When a codeword arrives at the receiver it initialises a counter to zero. It then examines each check bit, $k = 1, 2, 4, \dots$ in the way explained in the previous table, to see if it has the correct parity. If not, it adds k to the counter. If the counter is zero after all the check bits have been examined (i.e., they were all correct) the codeword is accepted as valid. If the counter is non-zero, it contains the position of the incorrect data bit. As shown in the following table, if check 1, 2 and 8 are in error, the inverted data bit is 11, because it is the only bit checked by bits 1, 2 and 8.

Example 6.3 Using the codeword of the previous example, but inverting the data bit in position 11:

	C_4	C_3	C_2	C_1
CHECK BIT POSITION	8	4	2	1
DATA BIT POSITION 3	-	-	0	0
DATA BIT POSITION 5	-	1	-	1
DATA BIT POSITION 6	-	1	1	-
DATA BIT POSITION 7	-	0	0	0
DATA BIT POSITION 9	1	-	-	1
DATA BIT POSITION 10	0	-	0	-
DATA BIT POSITION 11	0	-	0	0
EXCLUSIVE OR	1	0	1	1

This is all very impressive until it is realized that the Hamming code for a 7 bit code word will correct single bit errors only. The channel efficiency in terms of data bits transmitted, is 4 bits of data for every 7 bits transmitted, or 57%. To be able to correct 2 or more bit errors will require additional check bits which would further lower the efficiency.

However, there is a trick that can be used to permit Hamming codes to correct burst errors. A sequence of k consecutive codewords are arranged as a matrix, one codeword per row. Normally, the data would be transmitted row-wise, one codeword at a time, from left to right. To correct burst errors, the data should be transmitted one column at a time, starting with the leftmost column. When all k bits have been sent, the second column is sent, and so on. When the message arrives at the receiver, the matrix is reconstructed, one column at a time. If a burst error of length k occurs, 1 bit in each of the k codewords will have been affected, but the Hamming code can correct one error per codeword, so the entire block can be restored. This method uses kr check bits to make blocks of km data bits immune to a single burst error of length k or less.

Despite the low efficiency of forward error-correcting codes, such as the Hamming code, these codes do have advantages. When the connection is a simplex channel then there is no alternative, while on some communication systems involving long delays, such as on a satellite channel, the delay in requesting a repeat transmission may be so long that a forward error-correcting code may be more economical.

The simplest way of *detecting* single bit errors in a codeword is to add a parity bit as explained in the previous section. When blocks (or a frame) of codewords are being transmitted, there is an increased probability that a codeword (and hence the block) will contain a bit error. In this case an extension to the error-detecting capabilities obtained by the use of a single bit error can be achieved by using an additional set of parity bits computed from the complete block of codewords in the frame.

6.4.7 Block check characters

With this method, each codeword in the frame is assigned a parity bit as usual. This is called the *transverse or row parity*. In addition, an extra bit is computed from the bits in each individual bit position of the codewords in the whole frame. This is the *longitudinal or column parity*. The resulting set of parity bits for each column is referred to as the *block (sum) check character*. The example illustrated in Fig. 6.10 uses odd parity for the transverse or row bits and even parity for the column or longitudinal bits.

P	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	
0	0	0	0	0	0	1	0	=STX
1	0	1	0	1	0	0	0	
0	1	0	0	0	1	1	0	
0	0	1	0	0	0	0	0	
1	0	1	1	0	1	0	1	
0	1	0	0	0	0	0	0	
1	1	1	0	0	0	1	1	
1	0	0	0	0	0	1	1	=ETX
1	0	0	0	0	0	0	1	=BCC

Transverse (row) parity bits (odd)

Longitudinal (column) parity bits (even)

Direction of transmission

Frame contents

Figure 6.10: Example of block sum parity checking

It can be deduced from the example in that table that, although two bit errors in the same codeword will escape the row parity check, they will be detected by the corresponding column parity check. This is true, of course, only if no two bit errors occur in the same column at the same time as illustrated by the highlighted bits in the example. The probability of this occurring is will be much less than the probability of two bit errors in a single codeword occurring. Hence the use of a block sum check significantly improves error detection.

Parity checking and block parity checking are still best suited to applications in which *random single-bit* errors are present. When bursts of errors are present, however, a more rigorous method must be used. We examine such a method in the next section.

read forward

6.4.8 Cyclic Redundancy Coding

Although the above scheme may be adequate sometimes, in practice, another method is in widespread use: the *polynomial code* (also known as a *cyclic redundancy code* or CRC code). Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A k -bit message is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} to x^0 . Such a polynomial is said to be of degree $k - 1$. The high-order (leftmost) bit is the coefficient of x^{k-1} ; the next bit is the coefficient of x^{k-2} , and so on. For example, 110001 has 6 bits and thus represents a six term polynomial with coefficients 1, 1, 0, 0, 0, 1 and $F(x) = x^5 + x^4 + x^0$.

Polynomial arithmetic is done modulo 2. There are no carries for addition or borrows for subtraction. Both addition and subtraction are identical to EXCLUSIVE OR. For example:

$$\begin{array}{r}
 10011011 \\
 +11001010 \\
 \hline
 01010001
 \end{array}
 \quad
 \begin{array}{r}
 00110011 \\
 +11001101 \\
 \hline
 11111110
 \end{array}
 \quad
 \begin{array}{r}
 11110000 \\
 -10100110 \\
 \hline
 01010110
 \end{array}
 \quad
 \begin{array}{r}
 01010101 \\
 -10101111 \\
 \hline
 11111010
 \end{array}$$

Long division is carried out the same way as it is in binary except that the subtraction is done modulo 2, as above. A divisor is said "to go into" a dividend if the dividend has as many bits as the divisor.

When the polynomial code method is employed, the sender and receiver must agree upon a *generator polynomial*, $G(x)$, in advance. Both the high- and low-order bits of the generator must be 1. To compute the checksum for some message with m bits, corresponding to the polynomial $M(x)$ the message must be longer than the polynomial. The basic idea is to append a checksum to the end of the message in such a way that the polynomial represented by the checksummed message is divisible by $G(x)$. When the receiver gets the checksummed message, it tries dividing it by $G(x)$. If there is a remainder, there has been a transmission error.

The algorithm for computing the checksum is as follows:

1. Let r be the degree of $G(x)$. Append r zero bits to the low-order end of the message, so it now contains $m + r$ bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to $G(x)$ into the bit string corresponding to $x^r M(x)$ using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The result is the checksummed message to be transmitted. Call its polynomial $T(x)$.

$$T(x) = x^r M(x) - R(x)$$

Table 6.5 shows the calculation for a message 1101011011 and $G(x) = x^4 + x + 1$.

It should be clear that $T(x)$ is divisible (modulo 2) by $G(x)$. In any division problem, if you diminish the dividend by the remainder, what is left over is divisible by the divisor. For example, in base 10, if you divide 210278 by 10941, the remainder is 2399. By subtracting off 2399 from 210278, what is left over (207879) is divisible by 10941.

Now let us analyze the power of this method. What kinds of errors will be detected? Imagine that a transmission error occurs, so that instead of the polynomial for $T(x)$ arriving, $T(x) + E(x)$ arrives. Each 1 bit in $E(x)$ corresponds to a message bit that has been inverted. If there are k 1 bits in $E(x)$, k single-bit

$$\begin{array}{lcl}
 \text{eg } T(x) = 1011 & x^3 + x + 1 \\
 E(x) = 0100 & x^2 \\
 \hline
 T'(x) = 1111
 \end{array}$$

$$\begin{array}{l}
 T'(x) = \frac{T(x)}{G(x)} + \frac{E(x)}{G(x)} \\
 \quad \quad \quad \downarrow \quad \quad \quad \downarrow \\
 \quad \quad \quad \text{The remainder} \quad \text{remainder} \\
 \quad \quad \quad \text{this is 0} \quad \text{non-zero}
 \end{array}$$

Say data is 37
 sender and receiver agree on divisor 9
 All data sent must be ~~divisible~~ divisible by 9

$$\textcircled{S} \quad 9 \overline{) 37} = 4 \text{ remainder } 1 \quad \textcircled{P}$$

Send original message ~~and~~ ^{minus} remainder — ie send 36
 (remainder subtracted from number) ~~correct~~
 if answer divisible by 9, no error.

Message: 1 1 0 1 0 1 1 0 1 1
 Generator: 1 0 0 1 1
 Message after appending 4 zero bits: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

$$\begin{array}{r}
 \begin{array}{cccccccccccc}
 & & & & & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
 1 & 0 & 0 & 1 & 1 & | & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0
 \end{array} \\
 \begin{array}{r}
 1 \ 0 \ 0 \ 1 \ 1 \\
 \hline
 1 \ 0 \ 0 \ 1 \ 1 \\
 \hline
 0 \ 0 \ 0 \ 0 \ 1 \\
 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 0 \ 0 \ 0 \ 0 \ 1 \ 0 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 0 \ 0 \ 1 \ 0 \ 1 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 0 \ 1 \ 0 \ 1 \ 1 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 1 \ 0 \ 1 \ 1 \ 0 \\
 1 \ 0 \ 0 \ 1 \ 1 \\
 \hline
 0 \ 1 \ 0 \ 1 \ 0 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 1 \ 0 \ 1 \ 0 \ 0 \\
 1 \ 0 \ 0 \ 1 \ 1 \\
 \hline
 0 \ 1 \ 1 \ 1 \ 0 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 1 \ 1 \ 1 \ 0 \leftarrow \text{remainder}
 \end{array}
 \end{array}$$

Transmitted message: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Table 6.5: Calculation of the polynomial code checksum.

errors have occurred. A single burst error is characterized by an initial 1, a mixture of 0s and 1s, and a final 1, with all other bits being 0. Upon receiving the checksummed message, the receiver divides it by $G(x)$; that is, it computes

$$[T(x) + E(x)]/G(x).$$

The remainder from
 $T(x)/G(x)$ is always 0, so the result of the computation is simply

$$E(x)/G(x).$$

Those errors that happen to correspond to polynomials containing $G(x)$ as a factor will slip by unnoticed, but all other errors will be caught.

If there has been a single-bit error, then $E(x) = x^i$, where i determines which bit is in error. If $G(x)$ contains two or more terms, it will never divide $E(x)$, so all single-bit errors will be detected. *ie $G(x)$ must not have x^i as a factor*

If there have been two isolated single-bit errors,

$$E(x) = x^i + x^j, \text{ where } i > j.$$

Alternatively, this can be written as

$$E(x) = x^j(x^{i-j} + 1).$$

If we assume that $G(x)$ is not divisible by x , a sufficient condition for all double errors to be detected is that $G(x)$ does not divide $x^k + 1$ for any k up to the maximum value of $i - j$ (i.e., up to the maximum length). Simple, low-degree polynomials that give protection to long messages are known. For example, $x^{15} + x^{14} + 1$ will not divide $x^k + 1$ for any k below 32768. *if $G(x)$ does not have x^{k+1} as a factor, $E(x)$ will not be divisible by $G(x)$*

If there are an odd number of bits in error, $E(x)$ contains an odd number of terms, (e.g., $x^5 + x^2 + 1$, but not $x^2 + 1$). Interesting enough, there is no polynomial with an odd number of terms that has $x + 1$ as a factor in the modulo 2 system. By making $x + 1$ a factor of $G(x)$, we can catch all errors consisting of an odd number of inverted bits.

To see that no polynomial with an odd number of terms is divisible by $x + 1$, assume that $E(x)$ has an odd number of terms and is divisible by $x + 1$. Factor $E(x)$ into $(x + 1)Q(x)$. Now evaluate

$$E(1) = (1 + 1)Q(x).$$

Since $1+1=0$ (modulo 2), $E(1)$ must be 0. If $E(x)$ has an odd number of terms, substituting 1 for x everywhere will always yield 1 as result. Thus no polynomial with an odd number of terms is divisible by $x + 1$.

Finally, and most important, a polynomial code with r check bits will detect all burst errors of length $\leq r$. A burst error of length k can be represented by

$$x^I(x^{k-1} + \dots + 1),$$

where I determines how far from the right hand end of the received message the burst is located. If $G(x)$ contains an x^0 term, it will not have x^I as a factor, so if the degree of the parenthesized expression is less than the degree of $G(x)$, the remainder can never be zero.

If the burst length is $r + 1$, the remainder of the division by $G(x)$ will be zero if and only if the burst is identical to $G(x)$. By definition of a burst, the first and last bits must be 1, so whether it matches depends

on the $r - 1$ intermediate bits. If all combinations are regarded as equally likely, the probability of such incorrect message being accepted as valid is $\frac{1}{2}^{r-1}$.

It can also be shown that when an error burst longer than $r + 1$ bits occurs, or several shorter bursts occur, the probability of a bad message getting through unnoticed is $\frac{1}{2}^r$ – assuming that all bit patterns are equally likely. Three polynomials have become international standards:

$$\begin{aligned} \text{CRC} - 12 &= x^{12} + x^{11} + x^3 + x^2 + x + 1 && 180F \\ \text{CRC} - 16 &= x^{16} + x^{15} + x^2 + 1 && \text{000318005} \\ \text{CRC} - \text{CCITT} &= x^{16} + x^{12} + x^5 + 1 && (1021) \end{aligned}$$

The last polynomial corresponds to ITU-T Recommendation V.41.

All three contain $x + 1$ as a prime factor. CRC-12 is used when the character length is 6 bits. The other two are used for 8-bit characters. A 16-bit checksum, such as CRC-16 or CRC-CCITT catches all single and double errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997 percent of 17-bit error bursts, and 99.998 percent of 18-bit and longer bursts.

Although the calculation required to compute the checksum may seem complicated, it has been shown that a simple shift register circuit can be constructed to compute and verify the checksums in hardware. In practice, this hardware is nearly always used.

6.5 Exercises

Exercise 6.1 A voice line needs to transmit one byte of information every 125 μs . Assuming that the electrical-optic interface of a fibre optic system takes 625 ns to process one bit, how many voice lines can be multiplexed through the fibre using TDM?

Exercise 6.2 Can fibre be used to transmit analogue data? Explain.

Exercise 6.3 Explain why the SONET basic transmission rate (STS-1) is 51.84 Mbps.

Exercise 6.4 The probability that a bit is transmitted without error is e . A file of 1 MB needs to be transferred between two computers. The data link layer transmits frames whose payloads are only 512 bits long. Derive an expression for the average number of retransmissions required during the file transfer. You may assume there is no error clustering.

Exercise 6.5 What should the Hamming distance be of a code that can correct 3 errors?

Exercise 6.6 If we require a code that will correct any single error in an 8-bit word, how many check bits are needed?

Exercise 6.7 Using the Hamming code, what is the actual word sent if the data is 1001101?

Exercise 6.8 Assume we are using the Hamming code method. If we receive the word 1011111, is that word in error? *Given bit & code = 1 1 1 1 1 1 1 message = 1 1 1 1*

Don't have bandwidth and time to ask for re-transmitted.

Exercise 6.9 Explain why error detecting codes are used more frequently than error correcting codes. In a high speed channel of 620 Mbps with a propagation delay of 20 ms, would it be better to use an error detecting code or an error correcting one? Give reasons for your answer.

Exercise 6.10 The word 1001101 needs to be transmitted. Using a generator polynomial of $x^4 + x^2 + x + 1$, what is the actual transmitted message using Cyclic Redundancy Coding.

$$G(x) = 10111$$

$$r = 4$$

$$\frac{x^r M(x)}{G(x)} \Rightarrow$$

$$\begin{array}{r} 1011011 \\ 10111 \overline{) 1011011} \\ \underline{10111} \\ 0010001 \\ 10111 \\ \underline{10111} \\ 0011000 \\ 10111 \\ \underline{10111} \\ 011110 \\ 10111 \\ \underline{10111} \\ 010010 \\ 10111 \\ \underline{10111} \\ 00101 \end{array}$$

$$T(x) = Mx^r + R(x)$$

$$= 10011010101$$

$$R(x) = 0101$$

$$00101$$

Chapter 7

Data Link Protocols

7.1 Objectives of this Chapter

In this discussion of the protocols which apply at layer two of the OSI reference model, we shall cover the following topics.

1. We first learn about the general functions of link layer protocols and the types of fields found in a typical link protocol.
2. Link layer protocols are similar to many Wide Area Network protocols. We therefore use the opportunity to introduce general protocol design principles, in particular
 - Error control and management.
 - Window flow control, and
 - Piggybacking.
 - Pipelining including the ARQ techniques of Go-back N and Selective Repeat.
 - Positive acknowledgements.
3. We discuss the Synchronous Data Link Control (SDLC) protocol in considerable detail and conclude with
4. Two Internet link protocols, namely SLIP and PPP.

The *Data Link Layer (DLL)* (also known as the *Data Link Control (DLC)* layer) sits above the physical layer in the OSI model (Refer to Sec. 1.7 on page 12). The Data Link Layer is the first layer where the bits have *logical* significance. This layer supervises the flow of information between adjacent network nodes. It uses error detection or correction techniques to ensure that a transmission contains no errors. (In transmission, a bit could become corrupted, but the physical level protocol would not know it). If the data link layer detects an error, it can either request a new transmission or, depending on the implementation, correct the error. It also controls how much information is sent at one time: Too much and the network becomes congested; too little and receiving ends experience excessive waits.

At the DLL data are transmitted as *frames*, which consist of groups of bytes organized according to a specified format. The DLL marks the beginning and the end of each outgoing frame with unique bit patterns and recognises these patterns to define an incoming frame. It then sends error-free frames to the next layer, the network layer (see Chapter 9). These frames usually involve the use of special control characters and message headers with specified formats. Depending on the function to be performed, three distinct sets of protocols can be distinguished, namely, *device control protocols*, *data-link-control protocols* and *end-to-end data-format protocols*. These can be defined as follows:

1. A terminal or work station may have one or more input/output mechanisms, such as a keyboard, a printer, a display, etc. Each device has attributes, such as carriage return, line feed, tabs, coordinate positioning, new page, scroll, etc. Device Control refers to the sending of commands to activate or use these device attributes. Such commands are the formats for the device-control protocol.
2. Data Link Control (DLC) concerns the operation of a data link between one DCE and another DCE or between a DCE and a DTE. DLC protocols are needed to manage the operation of the link, which may be shared by multiple stations. The protocols regulate the initiation, checking and retransmission (if necessary) of each data transmission.
3. End-to-end data-format controls involve data-format indicators and headers to characterize the data being sent rather than its transportation. This category of controls includes, for example, the following:
 - Delimiters of text, to indicate the end of text or the separation of text from a header that describes the text.
 - Indicators for chaining together a group of data units.
 - Headers that identify the type of message being sent.
 - Indicators of whether or not a message obeys a prescribed format.

For these different protocols, the same control character came to mean different things. For example, a negative acknowledgement (NAK) might occur because of a line error *or* because of buffer unavailability. Since the end-user (usually an application) inserted these control characters, each application could assign its own conditions for the use of the control character. It thus became impossible to handle link controls at the link control level; all control characters had to be interpreted within the application. Thus, application programs were written to take advantage of the existing protocols. They contained diverse control functions that were both link- and device-dependent. Any change in a device or a line protocol required an application rewrite.

Although some of the older, confused protocols still persist, clearly defined higher (second) level protocols now exist with clearly defined functions. These are known as *Data Link Control protocols*, *Line Control procedures* or *Link Control procedures*.

Basically, the job of a DLC protocol is to:

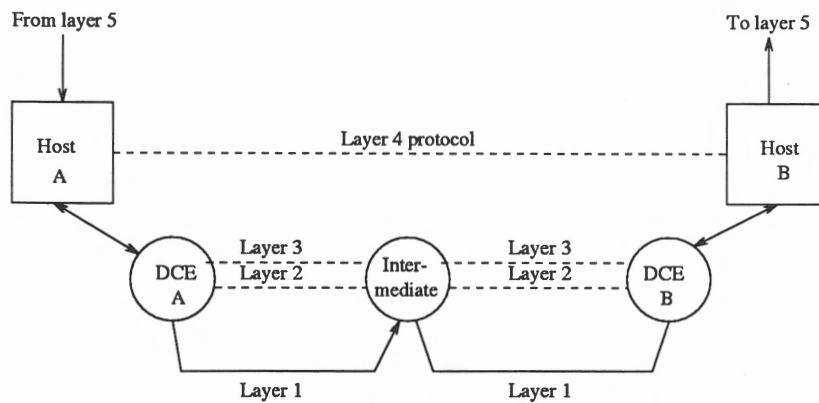


Figure 7.1: Conceptual model of Layer 2, 3 and 4 protocols

1. *establish and terminate a logical connection between stations*
2. *handle the orderly transfer of data between them*
3. *ensure message integrity in those transfers.*

We shall discuss data link control protocols mainly from the perspective of DCE-DCE or DCE-DTE communication.

7.2 Principles of Protocol Design

Before discussing particular DLC protocols we first discuss the design principles which one finds in any error control protocol operating in a Wide Area Network such as those comprising the Internet. Although we shall discuss the principles in the context of the DLL, they apply to a wide range of protocols at all levels of communication.

In the ISO multilevel protocol hierarchy (see Fig. 1.7 on page 13) the application process, in principle passes information (called PDUs; see Sec. 1.7) to Layer 6, which then gives it to Layer 5, the Session Layer, which gives it in turn to Layer 4, the Transport Layer. The Transport Layer software, splits the data up into messages if need be, and attaches a Transport Layer header to the front of each message. It then passes these messages to its DCE. The DCE attaches its own Network Layer headers to the front of each message, thus forming a DCE-DCE packet. The layer 3 DCE-DCE packet header might contain the source and destination addresses, routing information, etc. Eventually, the packet gets down to Layer 2, where a frame header and frame trailer are attached for transmission, and the frame is sent down to the Physical Layer 1 for transmission to the next DCE.

Our conceptual model, as shown in Fig. 7.1 is that a DTE A wants to send a long stream of data to DTE B. Later, we will consider the case where DTE B also wants to send data to DTE A simultaneously, but for the time being just consider simplex data transmission from A to B.

DTE A is assumed to have an infinite supply of data ready to sent and never has to wait for data to be produced. When the DCE asks for data, the DTE is always able to comply immediately. (This restriction, too, will be dropped later.) As far as the Layer 2 software is concerned, the packet coming from the DTE is pure data, every single bit of which is to be delivered to the other DTE. The fact that the receiving DTE may interpret part of the packet as a header is of no concern to the Layer 2 software.

When a DCE accepts a packet, it uses the packet to construct a frame. The frame usually contains certain control information in the *header* in addition to the message itself. The frame is then transmitted to the next DCE. The transmitting hardware computes and appends the checksum so that the DCE software need not worry about it. The polynomial algorithm discussed in Sec. 6.4.8 may, for example, be used for this purpose.

Providing a rigid interface between DTE and DCE greatly simplifies the software design, because the communication protocols and DTE protocols can evolve independently. It would be terrible if every little change to the lower-layer protocols required changing all the DTE operating systems. Since the DCE in most networks are generally identical pieces of hardware, changing the DCE software merely requires changing one program and then shipping it out to all the DCE sites (via the network itself, naturally). Changing the DTE protocol is much more complicated if, as is often the case, the DTEs are not all identical. Such a change requires modifying every operating system present on the network, a vast task.

When a frame arrives at the receiver, the hardware computes the checksum. If the checksum is incorrect (i.e., there was a transmission error), the DCE is so informed. If the inbound frame arrived undamaged, the DCE is also informed, so that it can do the necessary processing of the frame. As soon as the receiving DCE has acquired an undamaged frame, it checks the control information in the header, and if everything is all right, the packet portion is passed to the DTE. Under *no* circumstances is a frame header ever given to a DTE.

A generic frame header is composed of four control fields:

1. **KIND** field: tells whether or not there are any data in the frame, because some of the protocols distinguish frames containing exclusively control information from those containing data as well.
2. **SEQ** field: used for sequence numbers.
3. **ACK** field: used for acknowledgements.
4. **INFO** field: contains one or more packets (the **INFO** field of a control frame is not used).

The first three fields clearly contain control information, and the last may contain the data to be transferred.

7.2.1 Acknowledgements

In principle any channels may be unreliable and may lose an entire frame upon occasion. To be able to recover from such calamities, the sending DCE must start an interval timer or clock whenever it sends a frame. If no reply (called an *ACKNOWLEDGEMENT* or *ACK*) has been received within a certain predetermined time interval, the clock times out and the DCE receives an interrupt signal whereupon the frame is re-transmitted.

7.2.2 Flow Control

Since we assumed that the sending DCE has an infinite supply of data, it continues to send one frame after the other. One of the main problems we have to deal with is how to prevent the sender from flooding the receiver with frames faster than the latter is able to process it. In very restricted circumstances, e.g. synchronous transmission and a receiving DCE fully dedicated to processing the one input line, it may be possible to time the sender such as to keep it from swamping the receiver. Usually, however, each DCE will

have several lines to attend to, and the time interval between a frame arriving and it being processed may vary considerably. If the network designers can calculate the worst-case behaviour of the receiver, they can program the sender to transmit so slowly that even if every frame suffers the maximum delay, there will be no overruns.

A more general solution to this dilemma is to have the receiver *provide feedback* to the sender via the ACK control frame mentioned above. In other words, the receiver sends a control frame back to the sender which, in effect, gives the sender permission to transmit the next frame. After having send a frame, the sender is required to wait until the next control frame, i.e. acknowledgement arrives back. Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called *stop-and-wait* protocols.

1. Acknowledgment
2. Time out / Limited # of times.
3. Sequence #s
4. Used several times before we stop.

7.2.3 Sequence Numbers

Now let us consider the unfortunately all-too-realistic situation of a communication channel that makes errors. Frames may be either damaged or lost completely. We assume that when a frame is damaged in transit, however, the receiver hardware will detect this when it computes the checksum. When the frame is damaged in such a way that the checksum is nevertheless correct (an exceedingly unlikely occurrence), all protocols can fail, i.e., deliver an incorrect message to the DTE as we shall discuss next.

At first glance it would seem that a combination of the time-out and acknowledgement mechanism would solve the problem of transmitted errors. If a damaged frame arrived at the receiver, it would be discarded. After a while the sender would time out and send the frame again. This process would be repeated until the frame was finally correctly received and acknowledged.

This scheme has a fatal flaw in it however for the following reasons: Remember that it is the task of the communications subnet to provide error free, transparent communication between DTE's. DTE A gives a series of messages to its DCE A and the subnet must guarantee that the identical series of messages are delivered to DTE B by DCE B, in the same sequence. In particular, neither DTE has a way of knowing, under the present system, that a message made up of several packets, has been lost or duplicated, so the subnet must guarantee that no combination of transmission errors, no matter how unlikely, can cause an error message containing duplicate packets to be delivered to a DTE.

Consider the following sequence of events:

1. DTE A gives packet 1 to its DCE. The packet is correctly received at B and passed to DTE B. DCE B sends an acknowledgement frame back to DCE A.
2. Now the *acknowledgement frame* itself gets lost completely. It just never arrives at all. Life would be a great deal simpler if the channel only mangled and lost data frames and not control frames, but sad to say, the channel is not very discriminating.
3. DCE A eventually times out. Not having received an acknowledgement, it (incorrectly) assumes that its data frame was lost or damaged and sends the frame containing packet 1 again.
4. The duplicate frame also arrives at DCE B perfectly and is also unwittingly passed to DTE B. If A is sending a file to B, part of the file will be duplicated, i.e. the copy of the file made by B will be incorrect and the error will not have been detected. In other words, the protocol will fail.

Clearly, what is needed is some way for the receiver to be able to distinguish a frame that it is seeing for the first time from a retransmission. The obvious way to achieve this is to have the sender put a *sequence number* in the header of each frame it sends. Then the receiver can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.

7.2.4 Piggybacking

So far we have assumed that data were transmitted in only one direction. In most practical situations, there is a need for transmitting data in both directions. Typically, there will be several independent processes in DTE A that want to send data to DTE B, and in addition, several processes on DTE B that want to communicate with DTE A.

One way of achieving full-duplex transmission would be for each connection to have two separate communication channels. If this were done, we would have four separate physical circuits, each with a "forward" channel (for data) and a "reverse" channel (for acknowledgements). In both cases the bandwidth of the reverse channel would be almost entirely wasted. In effect, the user would be paying the cost of two circuits, but only using the capacity of one.

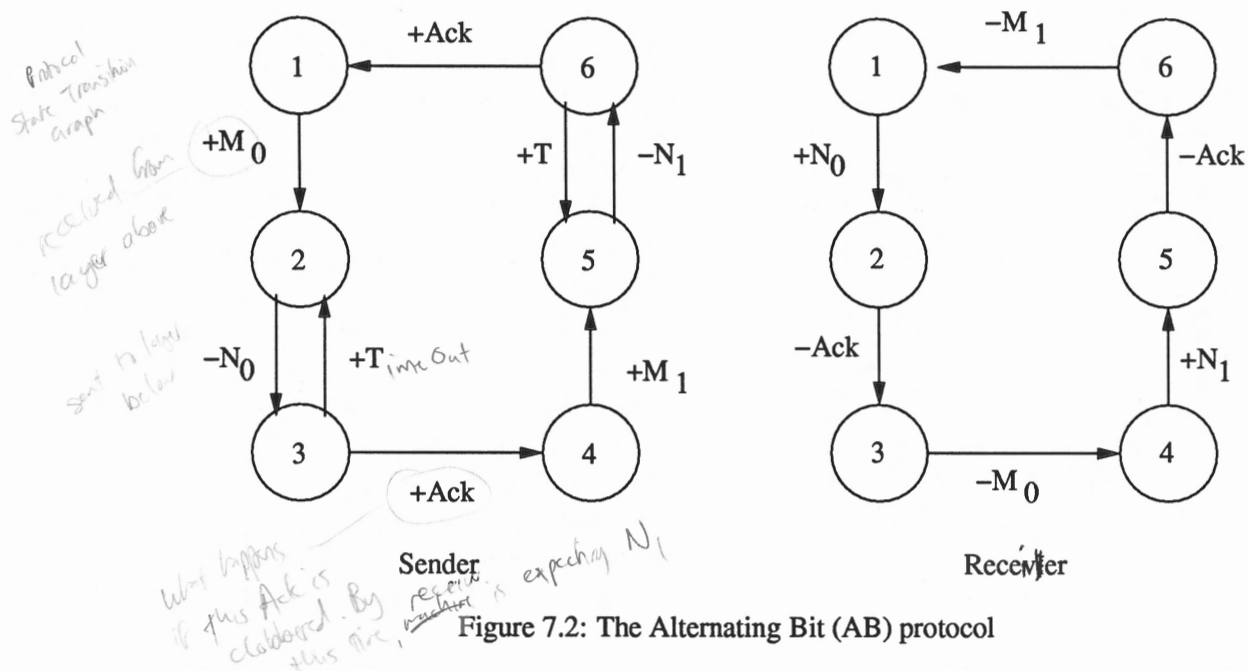


Figure 7.2: The Alternating Bit (AB) protocol

A better idea is to use the same circuit for data in both directions, so that data frames from A to B would be intermixed with the acknowledgement frames from B to A. By looking at the KIND bit in the header of an incoming frame, the receiver can tell whether the frame contains data or is an acknowledgement.

Although interleaving data and control frames on the same circuit is an improvement over having two separate physical circuits, yet another improvement is possible. When a data frame arrives at a DCE, instead of immediately sending a separate control frame, the DCE restrains itself and waits until the DTE passes it the next message. The acknowledgement is attached to the outgoing data frame (using the ACK field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is widely known as *piggybacking*.

The principal advantage of piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth. The ACK field in the frame header only costs a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum. In addition, fewer frames sent means fewer "frame arrived" interrupts, and perhaps fewer buffers in the receiver depending on how the receiver software is organized. In most protocols the piggyback field rarely costs more than a few bits in the frame header.

Piggybacking introduces a complication not present with separate acknowledgements. How long should the DCE wait for a data frame onto which to piggyback the acknowledgement? If the DCE waits longer than the sender's timeout period, the frame will be transmitted, defeating the whole purpose of having acknowledgements. If the DCE were an oracle and could foretell the future, it would know when the next DTE packet was going to come in, it could decide either to wait for it or send a separate acknowledgement immediately, depending on how long the projected wait was going to be. Of course, the DCE cannot foretell the future, so it must resort to some ad hoc scheme, such as waiting a fixed number of milliseconds. If a new DTE message arrives quickly, the acknowledgement is piggybacked onto it; otherwise, if no new DTE packet has arrived by the end of this time period, the DCE just sends a separate acknowledgement frame.

These principles are illustrated in the state transition graphs of the SENDER and RECEIVER using the *Alternating Bit (AB)* protocol illustrated in Fig. 7.2. In that figure a symbol "+" in front of a message indicates that that message is received at that point while a "-" indicates that the message is sent¹. The transitions $+m_0/+m_1$ in the SENDER state graph indicate the arrival of a message from the user (the network layer in our discussions) for transmission and similarly $-m_0/-m_1$ in the RECEIVER graph, the delivery of the corresponding message to the user at the receiving end. The term "alternating-bit" obviously derives from the single bit sequence number of the protocol.

7.2.5 Sliding Window Protocols

used for
 - used for error correction/control. Data integrity.
 - Also used to throttle down speed. Flow control.
 - Link management - pinging to see if the partner is up

A protocol which maintains a sequence number in the header of each frame it sends is often referred to as a *sliding window protocol*.

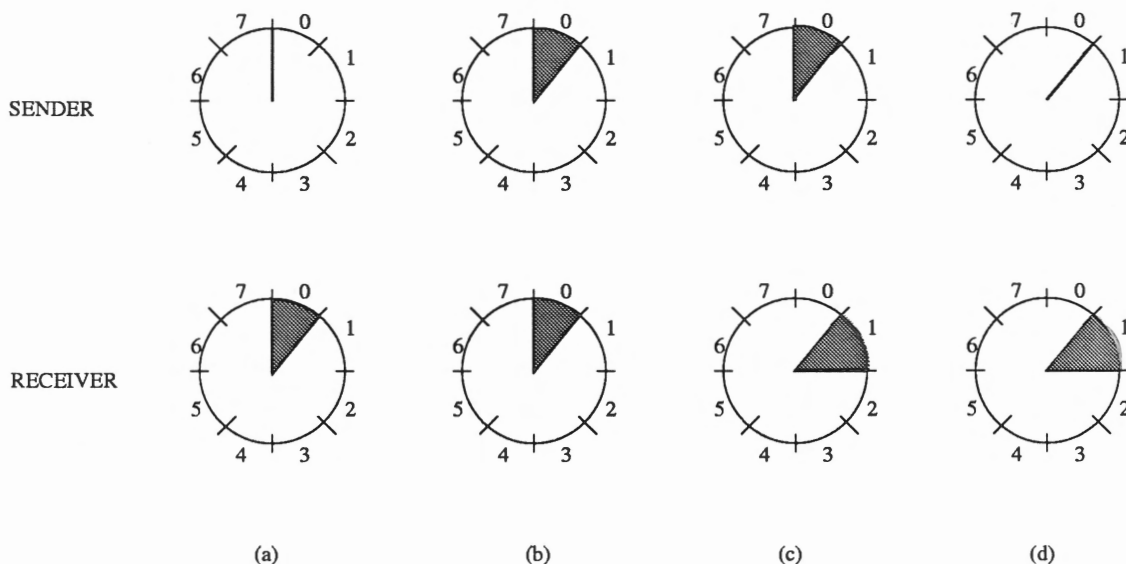


Figure 7.3: A sliding window of size 1, with a 3-bit sequence number. Part (a) initially (b) after the first frame has been sent (c) after the first frame has been received (d) after the first acknowledgement has been received – page 85

In all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually $2^n - 1$ so the sequence number fits nicely in an n bit field. The stop-and-wait sliding window protocol uses $n = 1$, restricting the sequence numbers to 0 and 1, but more sophisticated versions can use arbitrary n . These principles are illustrated in Fig. 7.3.

¹The symbol “r” and “s” are also used in the literature to indicate “receive” and “send” respectively.

To ensure that we do not receive duplicate frames the sequence numbers do not recycle in less than twice the window size. If window is size 8, the sequence #s will run from 0-15

The essence of all sliding window protocols is that at any instant of time, the sender maintains a list of consecutive sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the *sending window*. Similarly, the receiver also maintains a *receiving window* corresponding to frames it is permitted to accept. The sending window and the receiving window need not have the same lower and upper limits, or even have the same size.

Although this principle gives the DCEs more freedom about the order in which they will send and receive frames, the requirement still exists that the frames must be delivered to the destination DTE in the same order that they were passed to the source DCE by the source DTE. Moreover, the requirement still exists that frames between DCEs are sent in the order received.

The sequence numbers within the sender's window represent frames sent but as yet not acknowledged. Whenever a new packet arrives from the DTE, the corresponding frame is given the next highest sequence number, and the upper edge of the window is advanced by one. When an acknowledgement comes in, the lower edge is advanced by one. In this way the window continuously maintains a list of unacknowledged frames.

Since frames currently within the sender's window may ultimately be lost or damaged in transit, the sender must keep all these frames in its memory for possible retransmission. Thus if the maximum window size is n , the sender needs n buffers to hold the unacknowledged frames. If the window ever grows to its maximum size, the sending DCE must forcibly shut off the DTE until another buffer becomes free.

The receiving DCE window corresponds to the frames it may accept. Any frame falling outside the window is discarded without comment. When a frame whose sequence number is equal to the lower edge of the window is received, it is passed to the DTE, an acknowledgement is generated, and the window is rotated by one. Unlike the sender's window, the receiver's window always remains at its initial size. Note that a window size of 1 means that the DCE only accepts frames in order, but for larger windows this is not so. The DTE, in contrast, is always fed data in the proper order, regardless of the DCE's window size.

Until now we have made the tacit assumption that the transmission time required for a frame to arrive at the receiver plus the transmission time for the acknowledgement to come back is negligible. Sometimes this assumption is patently false. In these situations the long round-trip time can have important implications for the efficiency of the bandwidth utilization. As an example, consider a 50Kbps satellite channel with a 270 millisecond round-trip propagation delay. Let us imagine trying to send 1000-bit frames via the satellite. At $t = 0$ the sender starts sending the first frame. At $t = 20$ millisecond the frame has been completely sent. Not until $t = 135$ millisecond has the frame fully arrived at the receiver, and not until $t = 270$ millisecond has the acknowledgement arrived back at the sender, under the best of circumstances (no waiting in the receiver and a short acknowledgement frame). This means that on a stop-and-wait protocol, the sender was blocked during 270/250 or 96 percent of the time (i.e., only 4 percent of the available bandwidth was used). Clearly, the combination of a long transit time, high bandwidth and short frame length is disastrous in terms of efficiency.

The problem described above can be viewed as a direct consequence of the rule requiring a sender to wait for an acknowledgement before sending another frame. If we relax that restriction, much better efficiency can be achieved. Basically the solution lies in allowing the sender to transmit up to w frames before blocking, instead of just 1. With an appropriate choice of w the sender will be able to continuously transmit frames for a time equal to the round-trip transit time without filling up the window. In the example above, w should be at least 13. The sender begins sending frame 0 as before. By the time it has finished sending 13 frames, at $t = 270$, the acknowledgement for frame 0 will have just arrived. Thereafter, acknowledgement will arrive spaced by 20 millisecond, so the sender always gets permission to continue just when it needs it. At all times, 12 or 13 unacknowledged frames are outstanding. Put in other terms, the sender's maximum window size is 13.

7.2.6 Pipelining

This technique is known as *pipelining*. If the channel capacity is b bits/sec, the frame size l bits, and the round-trip propagation time R sec, the time required to transmit a single frame is l/b seconds. After the last bit of data frame has been sent, there is a delay of $R/2$ before that bit arrives at the receiver, and another delay of at least $R/2$ for the acknowledgement to come back, for a total delay of R . In stop-and-wait the line is busy for l/b and idle for R , giving a line utilization of $l/(l + bR)$. If $l < bR$ the efficiency will be less than 50 percent. Since there is always a finite delay for the acknowledgement to propagate back, in principle pipelining can be used to keep the line busy during this interval, but if the interval is small, the additional complexity is not worth the trouble.

Pipelining frames over an unreliable communication channel raises some serious issues. First, what happens if a frame in the middle of a long stream is damaged or lost? Large numbers of succeeding frames will arrive at the receiver before the sender even finds out that anything is wrong. When a damaged frame arrives at the receiver, it obviously should be discarded, but what should the receiver do with all the correct frames following it? Remember that the receiving DCE is obligated to hand packets to the DTE in sequence.

There are various approaches to dealing with errors in the presence of pipelining. One way, called *Go-back N*, is for the receiver simply to discard all subsequent frames, sending no acknowledgements as illustrated in Fig. 7.4. This strategy corresponds to a receive window of size 1. In other words, the DCE refuses to accept any frame except the next one it must give to the DTE. If the sender's window fills up before the timer runs out, the pipeline will begin to empty. Eventually, the sender will time out and retransmit all unacknowledged frames in order, starting with the damaged or lost one. This approach can waste a lot of bandwidth if the error rate is high.

Another general strategy for handling errors when frames are pipelined, called *Selective Repeat*, is to have the receiving DCE store all the correct frames following the bad one. When the sender finally notices that something is wrong, it just retransmits the one bad frame, not all its successors. If the second try succeeds, the receiving DCE will now have many correct frames in sequence, so they can all be handed off to the DTE quickly and the highest number acknowledged. This technique is also illustrated in Fig. 7.4.

Accepting frames out of sequence introduces certain problems not present in protocols in which frames are only accepted in order. We can illustrate the trouble most easily with an example. Suppose that we have a 3-bit sequence number, so that the sender is permitted to transmit up to eight frames before being required to wait for an acknowledgement. Simultaneously, the receiver's window allows it to accept any frame with sequence number between 0 and 7 inclusive. Assume that the transmitter has a message consisting of more than eight frames. Frames with sequence numbers 0 through 7 are therefore first sent to the receiver. These eight frames arrive correctly, the receiver sends a piggybacked acknowledgement for frame 7 (since 0 through 7 have been received), and delivers the frames to the DTE. The receiver window remains open to accept the next frames with sequence numbers 0, ..., 7.

It is at this point that disaster strikes in the form of a lightning bolt hitting the microwave tower, and wiping out the acknowledgements for frames 0, ..., 7 which have just been transmitted to the sender. Since the acknowledgement has been destroyed, the sender eventually times out and retransmits frames 0, ..., 7. When these frames arrive at the receiver, a check is made to see whether they are within the receiving window. Unfortunately, they are and these (duplicate) frames will be accepted and (erroneously) delivered to the DTE. Consequently, the DTE gets an incorrect message, and the protocol fails.

The essence of the problem is that after the receiver advanced its window, the new range of valid sequence numbers overlapped the old one. The following batch of frames might be either duplicates (if all the acknowledgements were lost) or new ones (if all the acknowledgements were received). The poor receiver has no way of distinguishing these two cases.

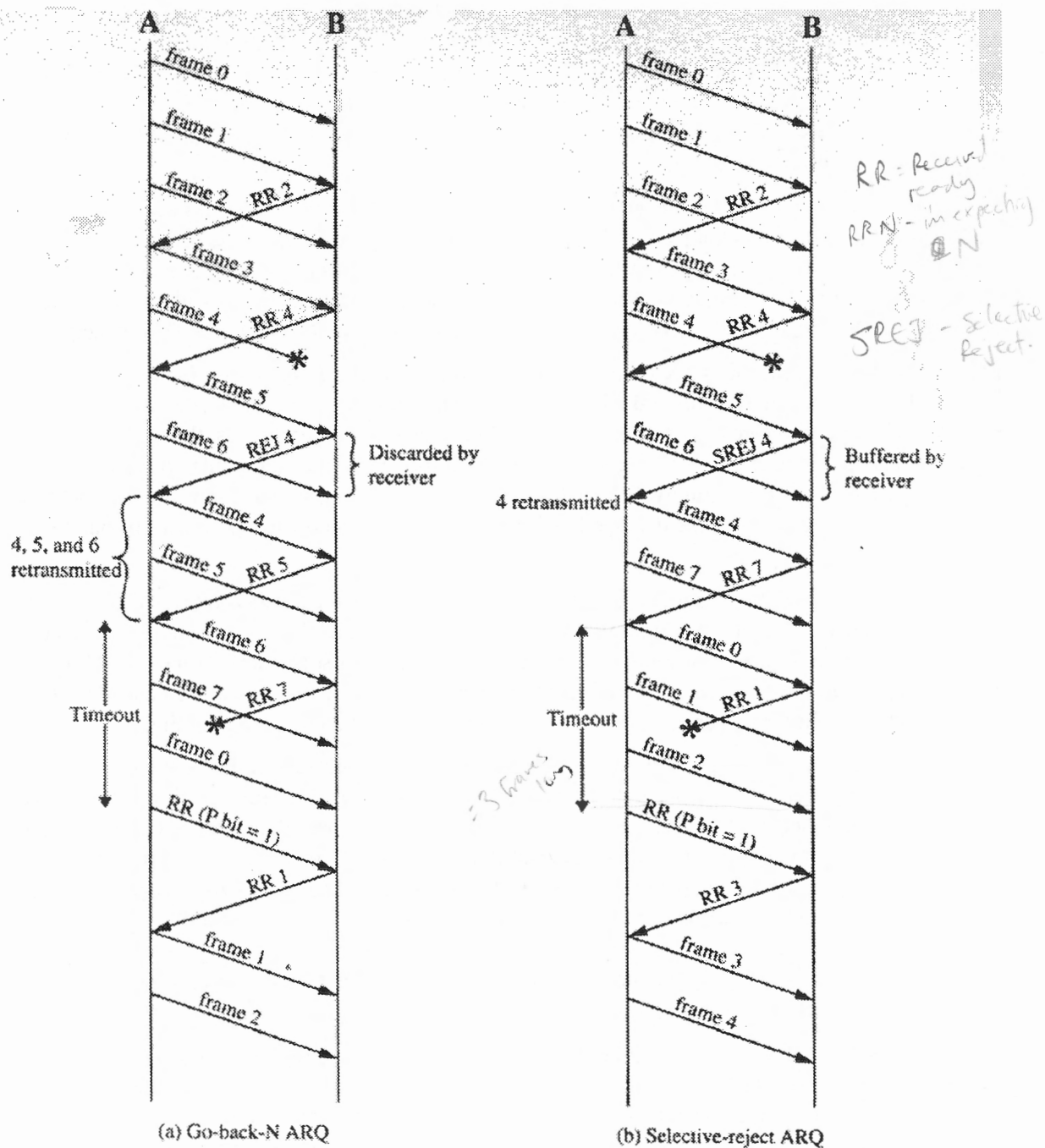


Figure 7.4: Pipelined ARQ protocols

The way out of this dilemma lies in making sure that after the receiver has advanced its window, there is no overlap with the original window. To ensure that there is no overlap, the maximum window size should be at most half the range of sequence numbers. For example, if 4 bits are used for sequence numbers, these will range from 0 to 15. Only eight unacknowledged frames should be outstanding at any instant. That way, if the receiver has just accepted frames 0 through 7 and advanced its window to permit acceptance of frames 8 through 15, it can unambiguously tell if subsequent frames are retransmissions (0 through 7) or new ones (8 through 15). In general, the window size will be $(MaxSeq + 1)/2$.

The protocols we have discussed so far are also collectively known as *Automatic Repeat Request* or *ARQ* protocols.

7.2.7 Positive Acknowledgements

The final general principle we should like to discuss is a more efficient strategy than having only time-outs to deal with errors. Whenever the receiver has reason to suspect that an error has occurred, it sends a *negative acknowledgement (NACK)* frame back to the sender. Such a frame is a request for retransmission of the frame specified in the NACK. There are two cases when the receiver should be suspicious: a damaged frame has arrived or a frame other than the expected one arrived (potential lost frame). To avoid making multiple requests for retransmission of the same lost frame, the receiver should keep track of whether a NACK has already been sent for a given frame. If the NACK gets mangled or lost, no real harm is done, since the sender will eventually time out and retransmit the missing frame anyway.

If the time required for a frame to propagate to the DCE, be processed there, and an acknowledgement returned is (nearly) constant, the sender can adjust its timer to be just slightly larger than the normal time interval expected between sending a frame and receiving its acknowledgement. However, if this time is highly variable, the sender is faced with the choice of either setting the interval to a small value, and risking unnecessary retransmissions, thus wasting bandwidth, or setting it to a large value, going idle for a long period after an error, thus also wasting bandwidth. If the reverse traffic is sporadic, the time to acknowledgement will be irregular, being shorter when there is reverse traffic and longer when there is not. Variable processing time within the receiver can also be a problem here. In general, whenever the standard deviation of the acknowledgement interval is small compared to the interval itself, the timer can be set "tight" and NACKs are not useful. Otherwise, the timer must be set "loose", and NACKs can appreciably speed up retransmission of lost or damaged frames.

7.3 Synchronous Data Link Control (SDLC)

The first of the DLC protocols we would like to mention is the *Synchronous Data Link Control (SDLC)* protocol because (not surprisingly) it illustrates all the principles described in the previous section.

IBM originally developed the *Synchronous Data Link Control (SDLC)* protocol, which ANSI modified to become *Advanced Data Communication Control Procedure (ADCCP)*, and ISO modified it to become the *High-level Data Link Control (HDLC)*. ITU-T then adopted and modified HDLC for its *Link Access Procedure (LAP)* as part of the X.25 network interface standard, but later modified it again to *LAPB*, to make it more compatible with a later version of HDLC.

The nice thing about standards is that you have so many to choose from; furthermore, if you do not like any of them, you can just wait for next year's model ...

SDLC and its related family of protocols are in fact *bit-oriented* in contrast to the mostly obsolete character-oriented protocols.

The reason that these protocols are referred to as “bit-oriented” is that each one allows data frames containing an arbitrary number of bits. The frame size need not be an integral multiple of any specific character size. Traditional protocols e.g., the Binary Synchronous Communication (BSC) protocol recognized the end of frame by some special character, such as ETB or ETX. Since bit-oriented frames do not have to contain an integral number of characters, a new method is needed to delimit the frame yet preserve data transparency i.e., allow arbitrary data, including the frame delimiter pattern. The SDLC frame format is illustrated in Fig. 7.5.

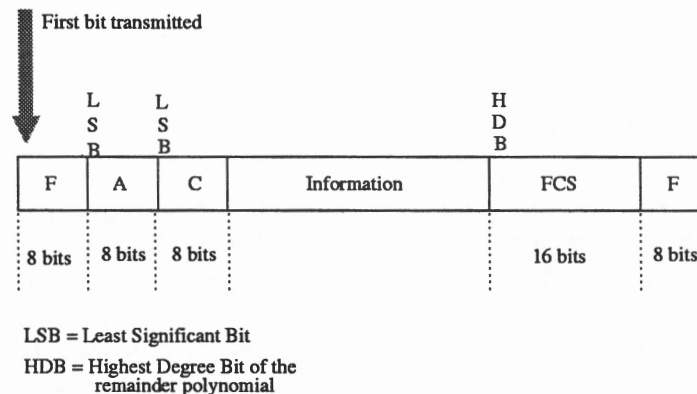


Figure 7.5: The SDLC frame format.

The various fields are as follows:

F *Flag*; used to delimit the frame.

A *Address-field*; primarily of importance on multidrop lines² where it is used to identify a terminal.

C *Control field*; used for sequence numbers, acknowledgements and other purposes as discussed below.

FCS *Checksum field*; a minor variation of the well known cyclic redundancy code using CRC-CCITT as the generating polynomial. The variation is to allow lost flag bytes to be detected.

The data itself is carried in the *Information field*.

SDLC is not a peer to peer *balanced protocol* but rather makes a distinction between the control or primary DTE and the secondary station or stations being controlled. In this way, a frame going from a primary station to a secondary station is called a *link command*; the frame going from a secondary station to the primary station is called a *link response*.

7.3.1 Flag field

Although the SDLC structure avoids the use of special characters within the transmission, SDLC does use a unique bit sequence for the start and termination of each variable-length frame.

The flag used by SDLC is a unique sequence of bits, namely, a zero, six one bits, and a zero (0111110). All receiving stations will continuously scan for this sequence, which signifies the beginning or end of a frame. SDLC uses the same flag byte for beginning and ending the frame as do HDLC and ADCCP.

²Not discussed in these notes.

To avoid an occurrence of this flag bit pattern anywhere else in a frame, a technique called “*bit stuffing*” is used. When transmitting, the sending station monitors the stream of bits being sent. Any time a contiguous string of five 1 bits occurs, there is a possibility of an unintentional flag developing; therefore, the sending station will automatically insert an extra 0 into the bit stream. Bit stuffing is illustrated in Fig. 7.6. This “0

```

Data set to be transmitted:      1111110011111011
Data set after bit-stuffing:     111110100111110011
Data set after bit-stuffing and
start-stop bits have been inserted: 0111111011111010011111001101111110
Thus, the frame to be transmitted is:

      01111110  111110100111110011  01111110
Start of Frame  User Data with Bit-Stuffing  End of Frame

```

Figure 7.6: An example of bit stuffing

bit insertion” prevents more than five contiguous 1s from ever appearing between the flags. Note that this applies to the address, control, and frame-check-sequence fields as well as to the information field, but not to the flag fields.

Then, when receiving, the station again monitors the bit stream. Whenever five consecutive 1s appear, the sixth bit is examined. If it is a 0, the receiving station deletes it from the bit stream prior to presenting the information to a higher level. If the sixth bit is a 1, then the sequence may be a flag or an error or, in the case of an SDLC loop configuration, a *go-ahead* signal. If the seventh bit is a 0, the station accepts the combination (01111110) as a flag; otherwise, it rejects the frame except in the case of a loop configuration.

Thus, using bit stuffing, the flag is kept unique to the beginning and end of a frame. Synchronization can be established based on the appearance of the flag, and the positional significance of each field can be established relative to the flag.

Recall that bit synchronization must be maintained during the entire frame, which may be quite long (limited by the buffers available at the receiving station and the ability to clear these buffers in time). Since bit synchronization is maintained by the pace of the bit stream itself, it is necessary that there be an adequate number of signal polarity transitions occurring periodically. This is achieved by a combination of the aforementioned bit stuffing and the use of codes such as the Manchester encoding (see Sec. 2.5.1 on page 34).

Thus, by a combination of techniques, there is assurance of establishment and maintenance of synchronization. The flag itself provides *message* synchronization and *character* synchronization, while the combination of bit stuffing and binary encoding ensures sufficient polarity transitions to maintain *bit* synchronization.

Some error (hardware or programming) may occur in the sending node during the transmission of a frame. In that case, it may be best to abort and ignore the partial frame that was sent. To do this (on any SDLC link other than a loop), the sending station ends the frame in an unusual manner by transmitting at least eight (but fewer than fifteen) contiguous 1 bits, with no inserted 0s. Receipt of eight contiguous 1 bits is interpreted by the receiving station as an abort.

A link is defined to be in an idle state when a continuous 1 state is detected that persists for 15 bit times.

7.3.2 Address field

At the link level it is necessary to allow for the possibility of more than one station, that is, more than one input/output point, on a given line. Stations may be cluster controllers or hosts or terminals.

The SDLC address field is one byte long, permitting up to 254 stations on a given link. An address field of all zeros is reserved, e.g., for testing purposes, and an address of all ones is reserved for a broadcast to "all-stations". Each station address pertains to only one station on one link, and it is completely separate from the network address of the logical unit. In fact, a message travelling between one DTE and another, may have to traverse many links along its way.

The station address always pertains to a secondary station on a particular link. The primary station is that one on the link that directs the multiple use of the link. The primary station also has responsibilities dealing with initial link activation, link deactivation, and recovery from error situations. All dialogues take place between the primary and secondary. The address field tells which secondary the frame is going to or coming from.

A secondary may, however, have more than one address for receiving. It will always have its own unique address for sending; but in addition, it may be part of one or more groups and have to recognize one or more group addresses. A special case is the "all-stations" address which all secondaries must recognize.

The ISO standard (HDLC) and the ANSI standard (ADCCP) include provisions for an extended address field. In *extended* mode, its address field would consist of a chain of bytes, with another address byte following so long as the first bit in any address byte is zero (the first bit of the address is the least significant bit). An example of an application where this mode might be useful would be the use of long aircraft IDs as the address, which could be displayed directly without address to ID translation.

7.3.3 Control field

The control field provides the personality of the data link control. It permits a single information transfer to serve multiple control functions. For example, one frame sent from the primary station to a secondary station may be used to

- send information from a primary station to a secondary station,
- acknowledge to the secondary that one or more specified frames, previously sent by the secondary, have been validly received, error-free,
- poll the station, authorizing it to send any frames that it has ready.

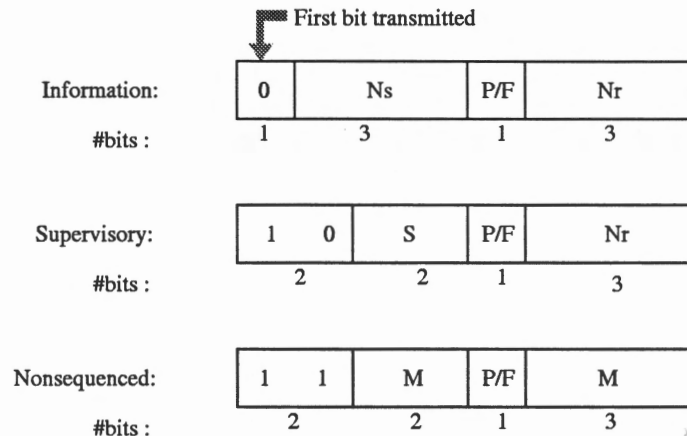
Similarly, a single frame sent from a secondary station to the primary station may be used to

- send information from the secondary to the primary,
- acknowledge to the primary that one or more specified frames, previously sent by the primary, have been accepted as error-free,
- indicate whether this is the final frame of a transmission or if more frames are to follow immediately.

There are three different formats for the control field, with a few bits assigned to identify which format is being used. The three formats, shown in Fig. 7.7 are the following.

- *Information transfer format (I)*, having full sequence control, with a send sequence count, N_s , and a receive sequence count, N_r , and (optionally) an information field.

- *Supervisory format (S)*, having a receive sequence count (N_r) but not a send sequence count, and used to manage the link in normal operation.
- *Nonsequenced format (NS)*, having no sequence counts at all, and used primarily for setting operating modes, exchanging identification, and other miscellaneous operations. (The HDLC and ADCCP terminology for this format is Unnumbered rather than Nonsequenced.)



Where N_s = Send Sequence Count (bit 2 = low order bit)
 N_r = Receive Sequence Count (bit 6 = low order bit)
 P/F = Poll Bit for Primary Station Transmission
 = Final Frame Bit for Secondary Station Transmission
 S = Supervisory Function Control Bits
 M = Modifier Function Control Bits

Figure 7.7: Three formats for the basic SDLC control field

Note that if the first bit of the control field is a 0, the I format is used; if the first bit is a 1, then the second bit tells whether an S format or an NS format is used. In all three formats 1 bit is used as a poll/final (P/F) bit. The primary station can use this bit to poll a secondary station. The poll bit also demands an acknowledgement (via the receive sequence count, N_r) or frames accepted by the secondary. The secondary station can use the same bit position to indicate the last frame to be sent (in response to a poll).

7.3.4 Acknowledgement and Retransmission.

Because duplex operation is a required capability, SDLC (like ADCCP and HDLC) adopted two independent sets of sequence numbers, one for each direction of flow. This permits the recovery of one flow to proceed independent of the sequence numbers of the other flow and avoids some possible timing ambiguities. Only information frames are sequence checked.

Two separate number counts are therefore maintained by each station. The *send sequence count* (N_s), operated modulo eight, provides a count for each information transfer frame that is transmitted from that station. The *receive sequence count* (N_r), also modulo eight, is incremented once for each valid, in-sequence, error-free, information frame that is received by that station. A secondary station, since it always transmits only to the primary, needs to keep only one pair of these sequence counts. The primary station, on the other hand, needs to keep one such pair for each secondary station to which it is transmitting. When one station sends its receive sequence count (N_r) to the other station, it will serve to indicate the next frame that is

Control (C)
Field

ITU-T (s. p. 155)

layer 1 - X-21
 layer 2 - LAP-B/HDLC
 link access protocol
 Balance
 layer 3 - X-25 packet layer

expected and to acknowledge all the frames received up to (but not including) the value indicated by the receive sequence number. Each subsequent *Nr* thus sent reconfirms that all preceding messages have been accepted. Multiple frames can be acknowledged in a single frame.

In the SDLC discipline, an acknowledgement is required at least once every seven requests. Otherwise, with modulo-eight, ambiguity in the response could result. The required verification also avoids buffer saturation. If too many frames are kept pending verification of their receipt, it could develop that too many buffers are full, leaving insufficient room for the receipt of data from the other end of the line. A lockup would then be possible.

If the receiving station has only a few buffers, then these must be emptied rapidly enough for the frames that are following. The sender, on the other hand, must have enough buffers to hold all frames sent until the receiver has acknowledged them. The waiting time for the acknowledgement is at least twice the delay in the transmission path, which is quite long in satellite transmissions.

It is important to realize that the performance of the link (in terms of the number of useful data bits transmitted per second) is dependent on the number of retransmissions that have to take place. The progress of a message from one DTE via one or more DCEs to its destination DTE is held up while the retransmission takes place. The transit times (hence, the response time) of that message, and of the others that might otherwise have gotten on the line sooner, are affected. Because of the uncertain reliability of the links, a very positive and definite acknowledgement is called for at the link level. Thus sequence numbers at the link level are applied to each of a possible series of links in the path between DTEs. It is intended to establish reliable transmission on each link without undue recourse to higher level protocols.

It is important to realize that the link-level sequence numbers are separate from the sequence numbers established for the dialogue between a source DTE and a destination DTE. Note that a given message with one sequence number may be segmented into several frames, each of which would then have a unique link sequence number for a particular link.

Note also that at the link level acknowledgements are mandatory, even though delayed. The send and receive counts must each be verified at least every seven transmissions. On the other hand, acknowledgements at the DTE-DTE level are optional and can be either none at all, exception response only, or definite response. Thus a wider range of response correlation techniques is needed at that higher level to match different application needs. This correlation is independent of the multiple link level correlations, which may be only on segments of higher-level requests.

7.3.5 Supervisory Format

In the *Supervisory format (S)*, two bits (the S-field) permit encoding of up to four control commands and responses for the purposes of controlling information flow. Three such commands are used by SDLC while a fourth is used by HDLC and ADCCP.

Type 0 is an acknowledgement frame (called *RECEIVE READY (RR)*) used to indicate the next frame (*Nr*) expected. This frame is used when there is no reverse traffic to use for piggybacking.

Type 1 is a negative acknowledgement frame (called *REJECT*). It is used to indicate that a transmission error has been detected. The *Nr* field indicates the first frame in sequence not received correctly (i.e., the frame to be retransmitted). The sender is required to retransmit all outstanding frames starting at *Nr* in Go-back N fashion.

Type 2 is *RECEIVE NOT READY (RNR)*. It acknowledges all frames up to but not including *Nr*, just like *RECEIVE READY*, but it tells the sender to stop sending. RNR is intended to signal certain temporary

problems with the receiver, such as a shortage of buffers, and not as an alternative to the sliding window flow control. When the condition has been repaired, the receiver sends a RECEIVE READY, REJECT, or certain control frames.

The type 3 command, *SELECTIVE REJECT*, is undefined in SDLC and LAPB, but is allowed in HDLC and ADCCP. It calls for the retransmission of only the one frame specified in *Nr* in Selective Repeat ARQ fashion. It is therefore most useful when the sender's window size is half the sequence space size, or less. Thus if a receiver wishes to buffer out of sequence frames for potential future use, it can force the retransmission of any specific frame using *SELECTIVE REJECT*.

7.3.6 Nonsequenced format

The third class of frame is the *Nonsequenced format* (NS). It is used for control purposes and the SDLC family at protocols differ considerably here. Five bits are available to indicate the frame function, but not all 32 possibilities are used.

The first command provided is *DISC* (*DISConnect*), that allows a to announce that it is going down (e.g., for preventive maintenance). There is also a command that allows a machine that has just come back on line to announce its presence and force all the sequence numbers back to zero.

Control frames may be lost or damaged, just like data frames, so they must be acknowledged too. A special control frame is provided for this purpose, called *NSA* (*Nonsequenced Acknowledgement*). Since only one control frame may be outstanding, there is never any ambiguity about which control frame is being acknowledged. No I field is permitted with the NSA response.

7.4 The Data Link Layer on the Internet

The Internet consists of individual machines (hosts and routers), and the communication infrastructure connecting them. We shall examine the data link protocols used on point-to-point lines on Internet, since most of the wide area infrastructure is built up from point-to-point leased lines.

Point-to-point communication is primarily used in two situations:

1. Organisations with one or more Local Networks (LANs, discussed in Chapter 8). Generally, all connections outside the LAN go through one or more routers that have point-to-point lines to other routers (which could connect to other LANs). These routers and their leased lines make up the communication subnet upon which Internet is built.
2. Home users who connect to Internet using modems and telephone lines.

In both cases, some point-to-point protocol is needed on the line for framing, error control and the other data link layer functions we have seen so far. The two main data link protocols used on the Internet are *Serial Line IP* (SLIP) and *Point to Point Protocol* (PPP).

7.4.1 Serial Line IP (SLIP)

SLIP is the older of the two protocols mentioned. It was originally devised to connect SUN Microsystems workstations to the Internet over a dial-up line using a modem. It is a very simple protocol.

The workstation simply sends raw packets over the line, with a special flag byte C0(hexadecimal) at the end for framing. If the end-of-frame character C0(hexadecimal) occurs inside the data packet, then the character sequence DBDC (hexadecimal) is sent instead (a form of character stuffing).

More recent versions of SLIP do some Transmission Control Protocol (TCP) and Internet Protocol (IP) (see Chapters 9) header compression by taking advantage of the fact that consecutive packets often have many header fields in common. Those fields which are similar to the previous packet are simply omitted. Furthermore, those fields which do differ are sent as increments to the previous value.

SLIP has various shortcomings:

1. It has no error detection/correction. This is left to the higher layers.
2. SLIP only works in conjunction with the Internet IP network protocol but, the Internet also supports networks which don't have IP as their native protocol.
3. IP addresses of both parties must be known in advance - neither address can be dynamically assigned during setup.
4. No authentication is provided.
5. SLIP is not an approved Internet standard, so many different (and incompatible) implementations exist.

7.4.2 Point-to-Point Protocol (PPP)

The Point-to-Point Protocol (PPP) was devised to avoid the SLIP shortcomings listed above. This was designed by the IETF (Internet Engineering Task Force) to be an official Internet standard. PPP handles error detection, supports multiple protocols, allows negotiation of IP addresses at connection time, provides for authentication and has many other improvements over SLIP. PPP is a multiprotocol framing mechanism suitable for use over modems, HDLC bit-serial lines, SONET and other physical layers.

In particular, PPP provides a:

1. Framing method that unambiguously delineates the end of one frame and the start of another. Error detection is included in the frame format.
2. Link control protocol for controlling lines (bringing them up, negotiating options and taking lines down). This is known as LCP (Link Control Protocol).
3. Way of negotiating network-layer options that is independent of the network-layer protocol to be used. A different NCP (Network Control Protocol) exists for each network layer supported.

The following is a typical scenario of an individual using PPP to connect to the Internet:

The PC workstation calls the router at the Service Provider via a modem. After the router's modem has responded and established a physical connection, the PC sends a series of LCP packets in the payload of one or more PPP frames. These elect the appropriate PPP parameters to be used.

Once these are agreed upon, a series of NCP packets are exchanged to configure the network layer. Typically, the PC wants to use the Internet TCP/IP protocol set, so it needs an IP address. There are not enough IP addresses to go around, so normally each ISP gets a block of these addresses and dynamically assigns one to each newly attached PC for the duration of its login session.

When the user ends his session, the NCP is used to “tear down” the network layer connection, and free up the IP address the PC was assigned. The LCP is then used to shut down the data link layer connection. The PC instructs the modem to hang up, thus releasing the physical connection.

The PPP frame format illustrated in Fig. 7.8 was chosen to resemble the HDLC (see Sec. 7.3) frame, since there was no reason to reinvent the wheel. All PPP frames begin with the standard HDLC *flag* byte 01111110

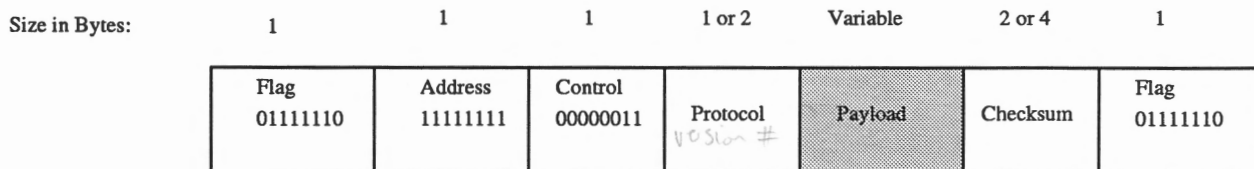


Figure 7.8: The PPP full frame format for unnumbered mode operation

(character stuffing is used if this appears in the payload field).

The *Address* field is next. It is set to the value 11111111 to indicate that any station may receive this frame. This avoids having to use data link addresses.

Next is the *Control* field, whose default value is 00000011. This indicates an unnumbered frame. In other words, PPP does not provide reliable transmission using sequence numbers and acknowledgements as the default, although reliable transmission using numbered mode can be specified.

Since the *Address* and *Control* fields are always the same in the default mode, LCP provides some header compression by allowing these fields to be dropped.

The *Protocol* field specifies what kind of packet is in the payload (remember, different protocols are supported). Codes are defined for LCP, NCP, IP, IPX and others.

The *Payload* field is of variable length, up to some agreed maximum. A default length of 1500 bytes is used if it is not specified (padding may follow the payload if need be).

Lastly, a *Checksum* field is appended, to provide for error detection.

7.5 Exercises

Exercise 7.1 What are the functions of a Data Link Control protocol?

Exercise 7.2 A channel has a capacity of 64Kbps and a propagation delay of 30ms. If we are using a stop and wait protocol, what is the minimum frame size we need in order for the efficiency to be above 50%?

Exercise 7.3 Consider a 128Kbps satellite with a 250 milliseconds round-trip propagation delay. Assuming a frame size of 1000 bits, what is the maximum window size for a sliding window protocol?

Exercise 7.4 Contrast SLIP and SDLC.

Exercise 7.5 Explain why pipelining improves the efficiency of a channel.

Exercise 7.6 In SDLC, the bit string 01111110 indicates frame boundaries. Explain how bit stuffing is used to preserve data transparency. Use the data string 1011 1111 0101 1111 0011 1111 1010 as an example.

Exercise 7.7 *Explain the term “piggybacking”. Include in your discussion any advantages and disadvantages of piggybacking.*

Exercise 7.8 *What is the purpose of the sliding window protocol? How is it implemented and what is the maximum size of the window using 7 bits to store the sequence number?*

Part III

Network Protocols

Chapter 8

Local Area Networks

8.1 Objectives of this Chapter

Local Area Networks are just what the name implies: Networks connecting workstations, file servers and print servers in a relatively (to WANs) restricted geographical area. The latter implies several simplifications compared to WANs. On completing this chapter, the reader will be familiar with the following:

1. Reasons for LANs and the implications for their protocols.
2. How LANs are classified on the basis of their topology and ways in which they access the common-transmission medium, known as Media Access Control (MAC) protocols.
3. We first learn about the Logic Link Control protocol which is used by all LANs
4. We study the various possible topologies, and classify them in random access techniques (the CSMA family) and
5. token passing or deterministic protocols.
6. Next we shall learn about the Ethernet protocol in detail, including
7. the principles of the FastEthernet protocol, a network with a 100Mbps capacity.
8. Amongst the token passing protocols, we shall learn about the Token Passing ring, originated by IBM, as well as
9. Token Bus protocol.
10. Finally, in the final section, we introduce the Wireless (IEEE 802.11) LAN protocol.

8.2 Introduction

When the computer equipment, workstations, file servers, print servers and so on which are to be connected are in close proximity such as in one building or a university campus, networking is a lot simpler than in the case of geographically spread out networks. The reasons are briefly as follows.

1. The entire network is under the control of one authority and decisions about which standards or technology to use are therefore centralised although the diversity of equipment and manufacturers may be much higher than in a WAN.
2. In all cases, because the propagation delay is microseconds, simple stop-and-wait is a good enough scheme and window flow control with all its error recovery complications, lost packets, etc is not needed.
3. Because the topologies, as we shall see, are such that each station receives its own transmission, explicit acknowledgements are not required and nor is flow control an issue.
4. Also, for the same reasons, a connectionless protocol is adequate.
5. Since the distances involved are small and confined to the premises, there is no need to involve a Service Provider or external cable company (traditionally the Telco) except to connect the network at a single point called the "gateway" to the Internet.

Although LANs have been around since the early 1970's, there is no general agreement on a precise definition for a Local Area Network. One useful definition, based on the bandwidth distance product, is the following:

Definition 8.1 BIT LENGTH OF A LAN. *The bandwidth of any communications link in bits per second multiplied by its length in meters defines the network in bit-meters per second. When this is divided by the propagation velocity¹ the result is the bit length of the network - that is, the number of bits in transit on the network at any given instant. In other words,*

$$(8.1) \quad \text{bit length} = (\text{capacity in bps}) \times \text{length} / \text{propagation velocity}$$

This figure can be used to define different types of networks. A conventional WAN, for example, has hundreds or thousands of bits in transit in any instant, whereas a computer bus has only a fraction of a bit ever "in transit". A local area network, because of its characteristic speeds and lengths, can be described as one whose bit-length falls between the two extremes.

The most widely used LANs are *Ethernet*, *token ring* and *token bus*. In a LAN, computer devices share a common transmission medium instead of being connected by point-to-point lines. Since LANs differ in transmission rates and by how they share access to the common links, different LANs are best suited to different applications. For example, Ethernet is adequate for low-load conditions when the delivery of packets is not subject to a hard time constraint. The token ring and token bus networks are preferable for high-load applications and also for hard time constraints.

¹ About 3×10^8 meters per second for fibre and 2.1×10^8 for copper.

8.3 Classification of LANs

With all the LANs around it is possible to distinguish them on the grounds of the way in which the various stations or nodes are linked, called the *topology* of the network and the way in which demand for the use of the connecting medium, or *bus* is arbitrated. The latter we call the *Medium Access Control (MAC)* protocol. We next discover what these terms mean.

8.3.1 Topologies

LANs function as non-centralised, distributed, multiple-access communications systems. This functional capability can be achieved by at least four basic topologies, *point-to-point*, *star*, *ring* and *bus*, either by themselves or in combination.

The cost and lack of versatility of the point-to-point topology, however, limit its application in local area environments; the star configuration requires central control with its associated vulnerability and is generally not suitable for a true LAN application although the most favoured LAN technology Ethernet 10BASE-T (see Sec. 8.6) uses this topology exclusively.

8.3.2 Common Bus Topology

Fig. 8.1 shows a *common bus topology* or simply a *bus topology*. The various nodes, which could be PCs, file servers, mainframes etc., communicate through a single bus constituting one or more parallel lines. The

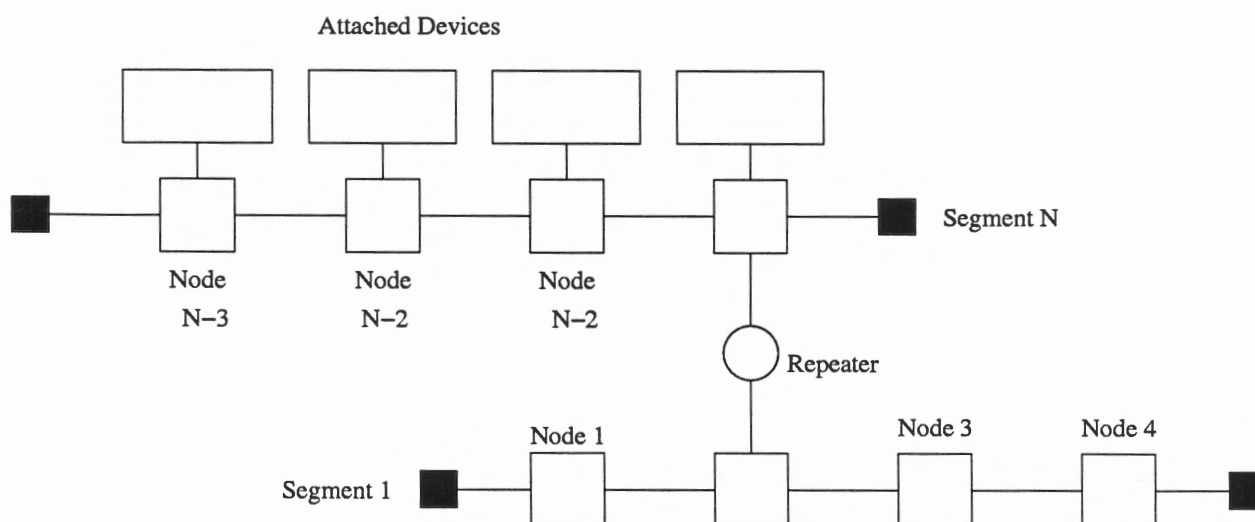


Figure 8.1: LAN Bus topology

most popular bus network used to be version 10BASE5 of the IEEE 802.5 standard, called *Ethernet* as we shall learn later in this chapter. It has since been replaced by its more popular 10BASE-T version which, while maintaining the same technology, is a star topology. In its original version, the common Ethernet bus was a 75Ω coaxial cable with although it is now possible to use optical fibre (or combinations of both).

Exam question: A bridge connects two networks where the protocols up to layer 2 are the same

8.3.3 Star Topology

Another common connecting arrangement is the *star topology* (shown in Fig. 8.2). It uses a central computer that communicates with other devices in the network. If a device wants to communicate, it does so only through the central computer. The central computer, in turn, routes the data to the destination. Centralization provides a focal point for responsibility, an advantage of the star topology.

However, the bus topology has some advantages over the star topology. The lack of central control makes the adding/removing of devices easy in a bus network. This is because no device may be aware of other devices. Also, the failure of the central computer in a star network will cause the whole network to fail. In a bus topology, the network will still function even if some of the devices fail.

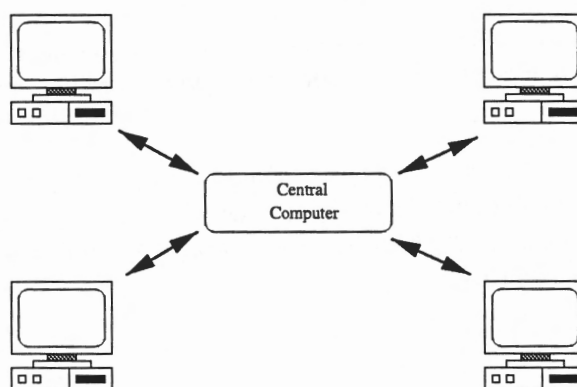


Figure 8.2: LAN Star topology

8.3.4 Ring Topology

In the *ring topology* of Fig. 8.3, devices are connected along a circle. Each one is physically connected to its neighbours on either side and but nobody else. If one wants to communicate with a device that is farther away, then a message must be sent in turn through each device that lies between the source and destination.

A ring network may be *unidirectional* or *bidirectional*. Unidirectional means that all transmissions travel in the same direction in which case a single conductor (or fibre) suffices. In this case each device can communicate only with one neighbour. Bidirectional means that data may travel in either direction, and a device can communicate with either of its neighbours which implies two signals paths, one in either direction. A disadvantage of the ring topology, is that when one station transmits to another station, all stations in between are involved. More time is spent relaying messages meant for others than in a bus topology (for example). Also, the failure of one station could cause a break in the ring which affects communication among other stations.

Many computer networks use combinations of the various topologies. using, possibly a common bus, sometimes called the *backbone*, which allows users to access mainframes and high volume or frequently accessed storage.

8.3.5 Media Access Control

LANs mostly use stations which are connected to a common bus or hub and as is the case for the bus between processors and memory on a computer motherboard, some arbitration mechanism is needed in

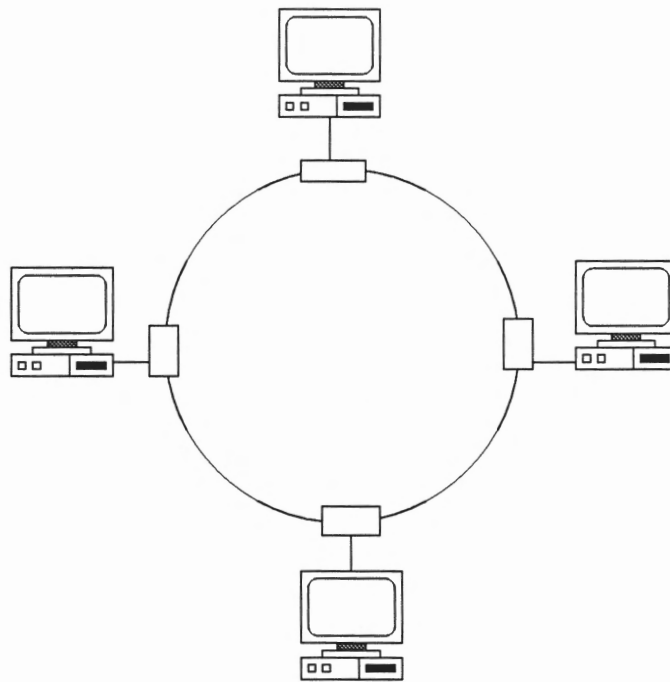


Figure 8.3: Ring topology.

order to determine who may use the bus next. One “orderly” way is to pass an agreed message or *token* around so that whoever has the token can transmit, something we understand from everyday life. Another way, alas all too common in everyday life as well, is for every station to simply transmit when it wants to and sorting out who gets the medium afterwards. The later are known as *Random Access Techniques* of which there are several as we now discuss.

8.4 Random Access Techniques

In the context of a LAN, just as when more than one person attempts to talk at the same time, when two or more nodes try to communicate at approximately the same time, their transmissions “collide”. In LAN terminology, we refer to this as a *collision*. Collisions are detected by nodes through the physical characteristics of the medium. More specifically, when a collision occurs, a channel’s energy level changes (e.g., the voltage level on copper-based media is usually at least twice as high as expected). During such times, the nodes’ signals become garbled. By monitoring the transmission line, nodes on the network are equipped to detect this condition.

What a node does when it detects a collision depends on the node’s MAC sublayer protocol. Similar to the way people might observe different protocols when contending to speak, nodes employ various protocols to transmit data in a shared media environment. For example, to minimize the occurrence of collisions, nodes might follow a protocol that requires them first to “listen” for another node’s transmission (somewhat incorrectly called a “*carrier*”, a term borrowed from radio terminology) before they begin transmitting data. We call these types of protocols (*Carrier Sense* protocols. Carrier Sense protocols require that before a node begins transmitting it must first listen to the medium to determine whether another node is transmitting data.

Carrier sense transmission systems also employ circuitry that requires the system to “listen” to every bit transmission going out and compare that to what is actually heard on the transmission medium. If they

match, wonderful. If they don't, something caused the bit to get hammered, and this means the transmission is garbled. How it is handled from there depends on the transmission framing method being used. There are 4 carrier sense protocols: *1-persistent CSMA (Carrier Sense Multiple Access)*, *Nonpersistent CSMA*, *CSMA with Collision Detection (CSMA/CD)* and *CSMA with Collision Avoidance (CSMA/CA)*. We'll describe these next.

8.4.1 1-persistent CSMA

When a node has data to transmit, it first senses the channel to determine if another node is transmitting. If the channel is not busy, then the node begins transmitting. If the channel is busy, then the node continuously monitors the channel until it senses an idle channel. Once it detects an idle channel, the node seizes the channel and begins transmitting its data. With this protocol, nodes with data to transmit enter a "sense and seize" mode they continuously listen for a clear channel and, once detected, begin transmitting data. This protocol is similar to a telephone with a multiple redial feature. If it detects a busy signal when the call is first attempted, the telephone repeatedly dials the number until a connection is finally established. The "1" in 1-persistent CSMA represents the probability (certain) that a single waiting node will be able to transmit data once it detects an idle channel. However, collisions can and do occur if more than a single node desires to transmit data at approximately the same time.

8.4.2 Non-persistent CSMA

can use exponential backoff algorithm.
This protocol is similar to 1-persistent CSMA, except a node does not continuously monitor the channel when it has data to transmit. Instead, if a node detects a busy channel, it waits a random period and re-checks the channel. If the channel is idle, the node acquires the channel and begins transmitting its data. If, however, the channel is still busy, the node waits another random period before it checks the channel again. Both 1-persistent CSMA and nonpersistent CSMA protocols eliminate almost all collisions. Except for those that occur when two nodes begin transmitting data nearly simultaneously. For example, node A begins transmitting data on a clear channel. A few microseconds later (which is not enough time for node B's sensing circuit to detect node A's transmission), node B erroneously declares the channel clear and begins transmitting its own data. Eventually, the two transmissions collide. 1-persistent CSMA involves a significant amount of waiting and unfairness in determining which node gets the medium. It is "selfish" because nodes can grab the channel whenever they feel like it. Non-persistent CSMA seems better because in its randomness there is fairness. In either case, though, there still remains the nagging problem of what to do about collisions. The next two CSMA protocols incorporate collision detection to provide a solution to the collision problem.

8.4.3 CSMA/CD

In this variant of either 1-persistent or nonpersistent CSMA, when a collision occurs the nodes (a) stop transmitting data, (b) send out a jamming signal, which ensures that all other nodes on the network detect the collision, (c) wait a random period of time, and then, (d) if the channel is free, attempt to retransmit their message. 1-persistent CSMA/CD is the MAC sublayer protocol used in Ethernet and IEEE 802.3 networks.

One question that frequently emerges when describing CSMA is the length of a random period. Well, it can be a long time, a short time, a moderate amount of time it's random, but it is also extremely critical. If the wait time is too short, collisions will occur repeatedly. If the wait time is too long, the medium could be in a constant idle state. Propagation delay also has an important effect on the performance of these

protocols. Consider the example given for nonpersistent CSMA: Node A senses an idle channel and begins transmitting. Shortly after node A has begun transmitting, node B is ready to transmit data and senses the channel. If node A's signal has not yet reached node B, then node B will sense an idle channel and begin transmitting. This will ultimately result in a collision. Consequently, the longer the propagation delay, the worse the performance of this protocol. Even if the propagation delay is zero, it is still possible to have collisions. For example, assume both nodes A and B are ready to transmit data, but they each sense a busy channel (a third node is transmitting at the time). If the protocol is 1-persistent CSMA, then both nodes would begin transmitting at the same time resulting in a collision. The specific amount of wait time is a function of the protocol.

8.4.4 CSMA/CA

This protocol is similar to CSMA/CD except that it implements collision avoidance instead of collision detection. As with straight CSMA, hosts that support this protocol first sense the channel to see if it is busy. If the host detects that the channel is not busy, then it is free to send data. What if every host connected to a LAN is not always able to sense each other host's transmission? In such instances, collision detection will not work because collision detection is predicated on nodes being able to hear each other's transmissions. This is the case with wireless LANs (WLANs). We cannot always assume that each station connected to a WLAN will hear each other station's transmissions. One direct consequence of this is that a sending node will not be able to detect if a receiving node is busy or idle. To address this issue, we replace CD with CA and use something called positive acknowledgment. It works like this:

The receiving host, upon receiving a transmission, issues an acknowledgment to the sending host. This informs the sending host that a collision did not occur. If the sending host does not receive this acknowledgement, it will assume that the receiving node did not receive the frame and it will retransmit it.

CSMA/CA also defines special frames called request to send (RTS) and clear to send (CTS), which further help minimize collisions. A sending node issues an RTS frame to the receiving node. If the channel is free, the receiving node then issues a CTS frame to the sending node. CSMA/CA is used in IEEE 802.11 (WLANs, see Sec. 8.8).

8.4.5 IEEE LAN Classification

One of the specific objectives of local area networks is to allow users with different types of data processing or terminal equipment to plug into the network with no concern for compatibility. All such equipment should therefore adhere to common protocols that will allow them to inter-operate.

Generally, LAN architectures span the two lower layers of the ISO OSI model, that is, they accommodate the functions of, and provide the services for, the Physical and Data Link layers of the model. Fig. 8.4 illustrates the LAN architecture proposed by the Institute for Electronic and Electrical Engineers (IEEE) standards committee and shows the corresponding OSI layers.

8.5 Logical Link Control (IEEE 802.2)

All that the IEEE 802 LANs and Metropolitan Area Networks (MANs, see Chapter ??) offer is a best-efforts datagram service. Sometimes this service is adequate. For example, for transporting IP packets, no guarantees are required or even expected. An IP packet can just be inserted into an IEEE 802 payload and sent on its way. If it gets lost, so be it.

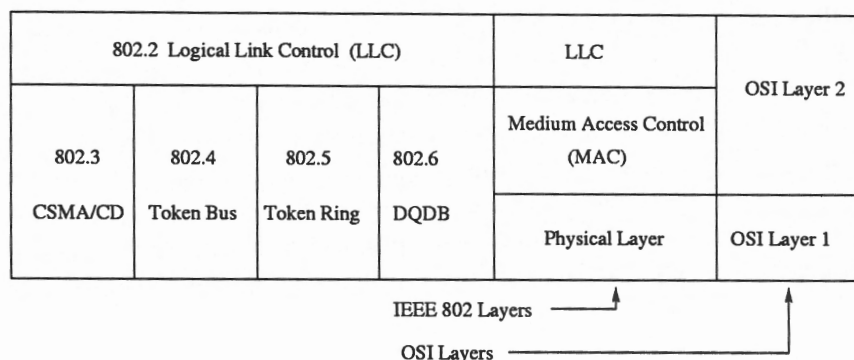


Figure 8.4: Correspondence between IEEE 802 layers and the OSI model

Nevertheless, there are also systems in which an error-controlled, flow-controlled data link protocol is desired. IEEE has defined one that can run on top of all IEEE 802 LAN and MAN protocols. In addition, this protocol, called LLC Logical Link Control), hides the differences between the various kinds of IEEE 802 networks by providing a single format and interface to the network layer. This format, interface and protocol are all closely based on OSI. LLC forms the upper half of the data link layer, with the MAC sublayer below it, as shown in Fig. 8.5.

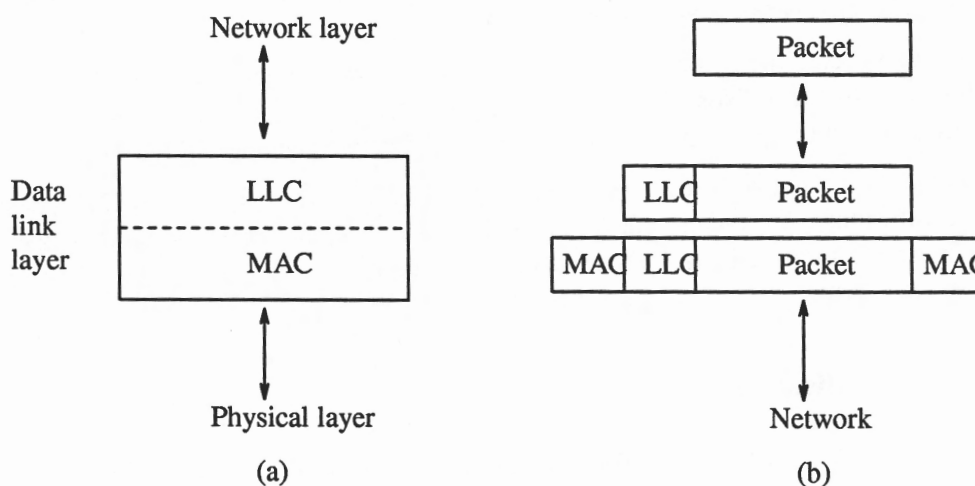


Figure 8.5: (a) Position of LLC and (b) Protocol formats

Typical usage of LLC is as follows: The network layer on the sending machine passes a packet to LLC using the LLC access primitives. The LLC sublayer then adds an LLC header, containing sequence and acknowledgement numbers. The resulting structure is then inserted into the payload field of an 802.x frame and transmitted. At the receiver the reverse process takes place.

LLC provides three service options: unreliable datagram service, acknowledged datagram service, and reliable connection-oriented service. The LLC header is based on the older HDLC protocol. A variety of different formats are used for data and control. For acknowledged datagram or connection-oriented service, the data frames contain a source address, a destination address, a sequence number, an acknowledgement number, and a few miscellaneous bits. For unreliable datagram service, the sequence number and acknowledgement number are omitted.

8.6 Ethernet

Networks based on the IEEE 802.3 random access protocol standard are by far the most prevalent. These bus networks use the media-access control protocol *Carrier Sense Multiple Access with Collision Detection (CSMA-CD)* referred to above.

There are several versions of IEEE 802.3. Here we will discuss only 10BASE5, 10BASE2, 1BASE5, 10BASE-T, and 10BROAD36. In these designations the first number (1 or 10) indicates the transmission rate in megabits per second. BASE indicates that the nodes transmit the information as baseband (digital) signals and BROAD identifies a broadband network that modulates an analog carrier signal with the digital frames². The number at the end on the designation (5, 2 or 36) is the maximum length of a segment of the networks a multiple of 100m. 10BASE-T designates a 10-Mbps baseband network that uses twisted pairs as the transmission medium. Thus, 10BASE5 is a 10-Mbps baseband network with segments of at most 500m.

The characteristics of these 5 versions are summarized in Table 8.1 The IEEE 802.3 networks use either

	10BASE5 ethernet	10BASE2 cheapernet	starLAN	10BROAD36 broadband	10BASE-T
Medium	Coaxial cable 50Ω-10mm	Coaxial cable 50Ω-10mm	Unshielded Twisted pair	Coaxial cable 75 Ω	Dual simplex Unshielded Twisted pair
Capacity	10 Mbps	10 Mbps	1 Mbps	10 Mbps	10 Mbps
Coding	Differential Manchester	Differential Manchester	Differential Manchester	DPSK	Differential Manchester
Segment Length	500 m	185 m	500 m	1 800 m	100 m
Network diameter	2.5 km	0.925 km	2.5 km	3.6 km	1 km
Nodes per segment	100	30			2
Collision Detection	Excess current	Excess current	2 active hub input	Transmission ≠ reception	Activity on receiver and transmitterSome

Table 8.1: IEEE 802.3 networks with 512 bit slot times, 96 bit gap times and jam signals lasting 32 – 48 bits

Unshielded Twisted Pair (UTP) or Coaxial cables while *dual* optical fibers are also possible. The Baseband networks use Differential Manchester encoding (see Sec: 2.5.1 on page 34). The Broadband version uses *Differential Phase Shift Keying (DPSK)* a modulation method that transmits the difference between successive bits using PSK modulation. Table 8.1 shows the maximum segment length, the maximum distance between nodes, and the maximum number of nodes per segment. The slot time is used to schedule retransmissions, and the gap time is the minimum time between two successive packets.

The original Ethernet protocol as patented by Metcalfe and Boggs [Met76] in 1976 before the IEEE802.3 standards, is almost identical to 10BASE5, except for minor differences in the frame structure. This version is based on a thick coaxial cable medium where the stations are arranged in a bus topology as illustrated in Fig. 8.6. We will base the following analysis of the CSMA/CD Medium Access Control algorithm on

²“Baseband” is another term for digital signal transmission of digital data. “Broadband”, in this context, means digital data send as modulated (DPSK) analog signals. Alas, as we know, “broadband” is also used to refer to networks with a capacity generally exceeding 100 Mbps.

this configuration and the CSMA/CD frame format illustrated in Fig. 8.7. In that configuration, stations are connected to the cable via transceivers and a transceiver connector; transceivers are spaced 2.5 meters apart to prevent signal interference. The actual physical connection of a transceiver, involves penetrating the cable using a so-called “vampire tap”. Each end of the cable is terminated with a 50Ω resistor, and one end of the cable must be at ground potential.

The transceiver converts the electrical signals on the cable into binary values for the interface board. To receive packets, this conversion is performed by a receiver circuit in the transceiver that measures the voltage on the cable. The circuit outputs a signal representing the binary value 1 when the voltage exceeds a fixed threshold and a signal representing 0 otherwise. As a result the interface board receives the Manchester-encoded bits transmitted on the cable. The interface board decodes the bits, stores them in its buffer, and informs the node when a full packet has been received.

To transmit a packet, the node copies it into a buffer on the interface board which waits for the transceiver to indicate when the coaxial cable is idle (the “Carrier Sense” part), i.e., when its receiver circuit does not detect any activity. The interface board then delivers the Manchester-encoded bits of the packet to the transceiver. The transceiver contains a current source that is switched on and off by the binary values of the Manchester code and that injects current with an average value between 18mA and 24mA into a coaxial cable. While a transceiver is transmitting, it monitors the current flowing through the cable by measuring the average voltage across the cable. If that average value shows that the current is larger than 24mA, then the transceiver knows that another transceiver is also injecting some current and it informs the interface board that a *collision* is occurring (the “Collision Detection” part).

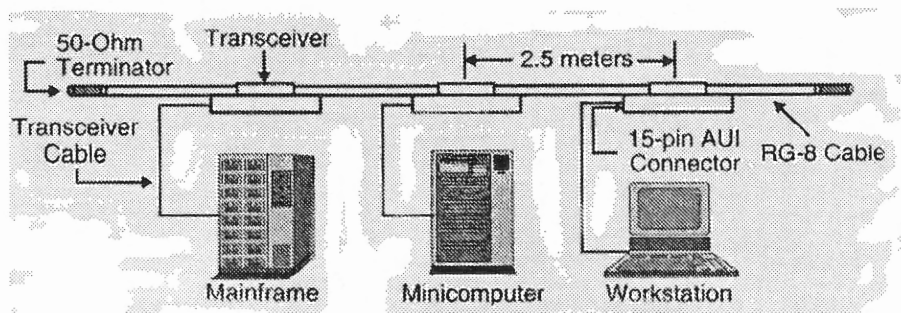


Figure 8.6: Ethernet 10BASE5 topology

When the interface board learns of the collision, it stops transmitting the packet and transmits a sequence of 32 to 48 randomly chosen bits, called the *jam signal*. The objective of the jam signal is to quarentee that all nodes become aware of the collision. The interface board then reschedules a retransmission of a packet after a random time. The interface board contains an integrated circuit that calculates the cyclic redundancy code bits to detect transmission errors.

In summary, the Ethernet MAC protocol (CSMA-CD) illustrated in the flow chart in Fig. 8.8 specifies that a node with a packet to transmit must proceed as follows:

1. Wait until the channel is idle.
2. Transmit and listen while transmitting.
3. In case of a collision, stop the packet transmission, transmit a jam signal, and then wait a random delay and go to 1.

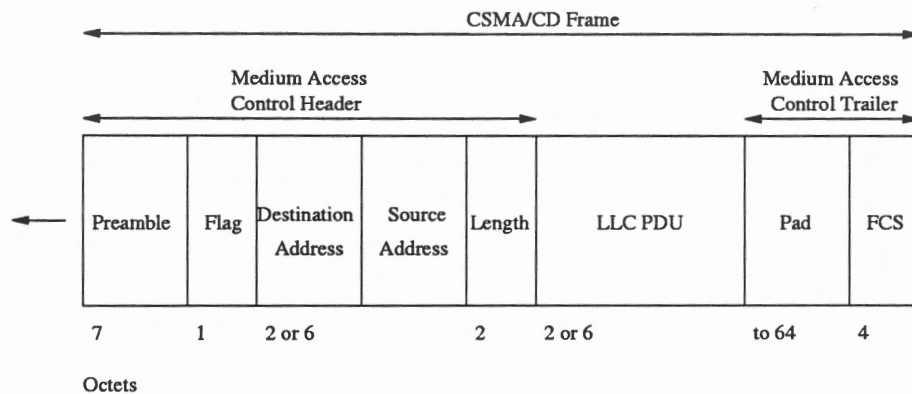


Figure 8.7: CSMA/CD frame format

The protocol stops the transmission after 16 successive collisions. The size of the random delay after a collision is selected according to the *binary exponential backoff algorithm*: if a packet has collided n successive times, where $n < 16$, then the node chooses a random number K with equal probability from the set $\{0, 1, 2, 3, \dots, 2^m - 1\}$ where $m := \min \{10, n\}$; the node then waits for $K \times 512$ bit times³. Thus after the first collision the nodes choose the random delay 0 or 512 bit times, with equal probabilities. After two successive collisions, the delay is equally likely to be 0, 512, 1024 or 1536 bit times. After three successive collisions, it is equally likely to be 0, 512, 1024, 1536, 2048, ..., or 3584 bit times. The rationale for using this method of selecting the value for the delay is that it quickly reduces the chances of repeated collisions by spreading out the range of waiting times.

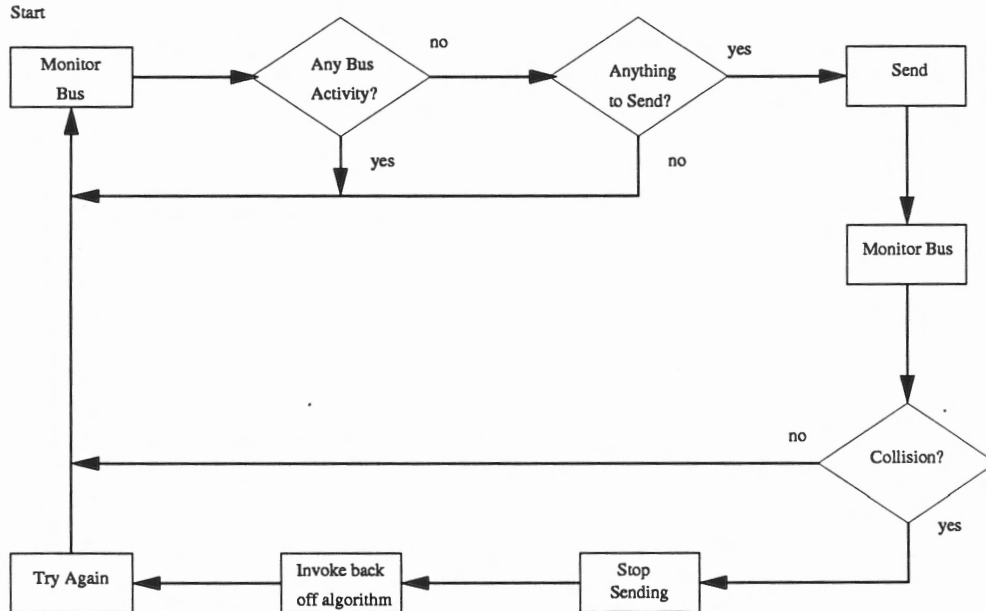


Figure 8.8: CSMA/CD MAC process

We saw that a node detects a collision while transmitting when its receiver measures an excessive current on the cable. The nodes that are not transmitting detect a collision when they observe a packet shorter than 500 bits, which is shorter than the minimum length of a valid frame (544 bits). A packet with fewer than

³At 10 Mbps, one bit time = 10^{-7} seconds.

500 bits appears on the cable when a collision occurs because the transmission time of a packet that collides cannot exceed the maximum collision detection time as shown in Fig. 8.10, which is twice the maximum end-to-end propagation time. The propagation time is determined by the cable length and the delays in the repeaters. The specifications of Ethernet result in a maximum collision detection time of 450 bit times. Consequently, the length of a packet that collides is at most 450 bits plus the maximum length of the jam signal (48 bits).

By far the most popular version of Ethernet is 10BASE-T illustrated in Fig. 8.9. A typical 10BASE-T wiring

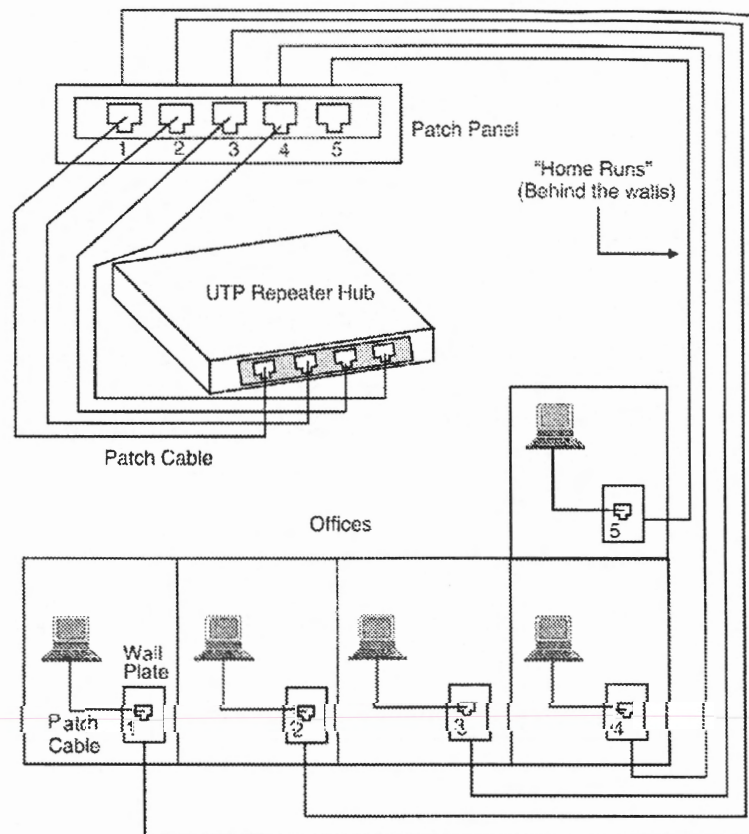


Figure 8.9: Popular networking configuration

scheme involves a centralised hub located in a *wiring closet* and lengths of cable installed between a *patch panel* (or *distributor board*) and a cable conduit plate in the end user office. In the trade this length of cable is referred to as the "*home run*". Each end of these home runs is then "punched down", usually run behind ceiling panels between the patch panel and the connector plate in the user's office. Patch cable is used to connect ports on the patch panel to the hub and workstations to the connector plate. The total length of all the cable from hub to patch panel, home run, and patch cable from conduit plate to workstation, may not be more than 100 meters.

Although patch panels provide a certain level of convenience, they are prone to noise. Nevertheless patch panels are useful in many situations. For example, note in Fig. 8.9 that port 5 on the patch panel is wired to office 5, but that workstation does not have a connection to the hub. Connectivity to office 5 can be provided though, by temporarily disconnecting any of the patch panel cables from one of the ports and re-connecting it to port 5.

8.6.1 Ethernet Performance

Next we examine the performance of Ethernet under conditions of heavy and constant load with k stations always ready to transmit. We use Fig. 8.10 to explain our analysis.

At the point marked t_0 a station in that figure has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision. Each will detect the collision, abort its transmission, wait a random period of time and try again, assuming that no other station has started transmitting in the meanwhile. Our model for Ethernet will therefore consist of alternating contention and transmission slots, with idle periods occurring when all stations are quiet (e.g. for lack of work). If each station transmits during a con-

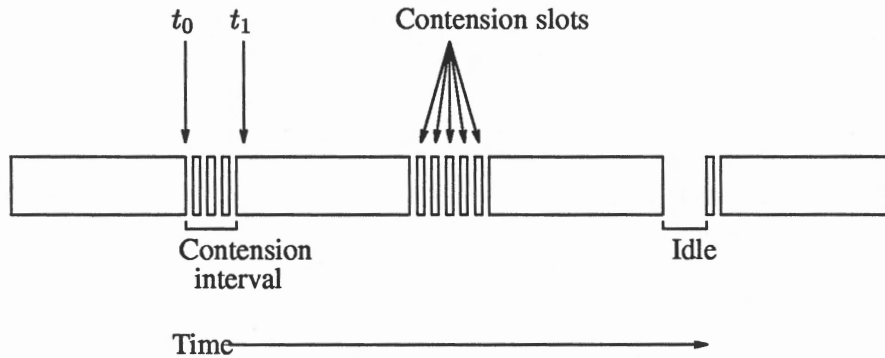


Figure 8.10: Ethernet states: contention, transmission, or idle.

tention slot with probability p , the probability Π , that some station acquires the ether during that slot is

$$\Pr\{\text{successful transmission}\} = \Pr\{\text{one station transmits}\} \times \Pr\{(k-1) \text{ stations do not transmit}\}$$

That is,

$$(8.2) \quad \Pi = kp(1-p)^{k-1}$$

which has a maximum value for $p = 1/k$. Also $\Pi \rightarrow 1/e$ as $k \rightarrow \infty$. The probability that the contention interval has exactly j slots in it is the same as the probability that for j slots no-one acquires the ether, but that it is acquired during the $(j+1)$ st slot, i.e., this probability is given by

$$(8.3) \quad \Pi(1-\Pi)^j$$

Hence the mean number of slots per contention is given by

$$(8.4) \quad \sum_{j=0}^{\infty} j \Pi (1-\Pi)^j = \frac{1-\Pi}{\Pi}$$

Since each slot has a duration 2τ , the mean contention interval ω is given by

$$(8.5) \quad \omega = 2\tau \left\{ \frac{1}{\Pi} - 1 \right\}.$$

Assuming optimal p , the mean number of contention slots is never more than e , so ω is at most

$$(8.6) \quad \omega = 2\tau(e-1) = 3.4\tau.$$

If the mean frame takes T seconds to transmit, when many stations have frames to send, the channel efficiency is given by

$$(8.7) \quad T/(T + 3.4\tau)$$

Here we see where the network diameter, the distance between the two remotest stations on the network enters into the performance figures. The longer the diameter, the longer the contention interval, the lower the channel efficiency.

8.6.2 Fast Ethernet

A few years back it became obvious that new operating systems, faster processors and larger disk and memory capacities result in network traffic that require capacities in excess of 10Mbps of the standard Ethernet. In addition increased Web traffic, either on the Internet or company intranets have given rise to multimedia traffic rich with high-resolution graphics, audio, video and voice. On top of that client-server computing and netcentric computing has taxed first-generation Ethernet technology.

To increase the capacity of an Ethernet network, IEEE increased the data transmission speed of the traditional IEEE 802.3 standard first to 100 Mbps and then to 1 Gbps. In June 1995, this new technology became the IEEE 802.3u standard and was given the specification 100BASE-T. Unlike 10BASE-T though, Fast Ethernet has 3 different media specifications: 100BASE-TX, 100BASE-T4 both using twisted-pair cable and 100BASE-FX which uses fibre-optic cable.

At the Data Link Layer, Fast Ethernet is unchanged from its 10 Mbps counterpart. The frame format, the minimum and maximum frame sizes and the MAC address format are all identical to conventional Ethernet. Most importantly, Fast Ethernet uses exactly the same media access method, namely CSMA/CD. This provides a simple and seamless migration path for users of 10 Mbps Ethernet to Fast Ethernet.

Table 8.1 summarises the 3 Fast Ethernet specifications with a brief description following that.

Name	Cable	Maximum Segment Length	Coding	Transmission
100BASE-T4	4-pair CAT 5 UTP	100 m	8B/6T	Half-duplex
100BASE-TX	2-pair CAT 3,4 or 5 UTP	100 m	4B/5B	Full-duplex
100BASE-FX	Dual Multimode Fibre	2000 m	4B/5B	Full-duplex

100BASE-T4 This specification uses inexpensive 3 UTP cable and a signalling speed of 25Mhz, only 25 percent faster than the standard IEEE 802.3 20 Mhz. To achieve the necessary bandwidth, 100BASE-T4 requires 4 twisted pairs, one from the station to the hub, one from the hub to the station and the other two are switchable to the current transmission direction. It uses a coding technique which maps 8 bit data blocks to a specific code group consisting of 6 symbols, the so-called 8B/6T method, to ensure synchronisation.

100BASE-TX Uses CAT 5 UTP and is therefore simpler since the wires can handle clock rates up to 125 MHz and more. Only two twisted pairs per station are used, one to the hub and one away from it. Rather than use the Manchester encoding scheme of conventional Ethernet, it uses a scheme called 4B/5B encoding which we discuss in detail in the second volume of these notes. 100BASE-TX is thus fully duplex.

100BASE-T4 This option uses two strands (how else, since fibre is uni-directional) of multimode fibre, one for each direction of transmission, so this option is also full-duplex with 100 MBps in each direction. In this case the distance between the station and the hub can be as much as 2 Km.

As a final note it is important to note that Fast Ethernet must be configured as a star topology, i.e., each station has to be connected to the hub. Traditionally Ethernet used a bus topology as we learned.

8.7 Deterministic or Token Passing Protocols

Unlike random access protocols, token passing protocols do not involve collision detection, but rely instead on granting stations on the network permission to transmit. Permission is granted in the form of a special control frame called a *token*. The underlying principle of token passing protocols is simple: The station that has the token may use the medium. Since possession of the token controls access to the medium, there is no contention and there are thus no collisions. The absence of contention also implies an absence of collision. Thus, token passing schemes are both contention-free and collision-free protocols.

The IEEE has standardised two LAN protocols using token passing. They are the Token Ring, IEEE 802.5 discussed here and IEEE 802.4, Token Bus discussed in the next section. Although not an IEEE standard, Fibre Distributed Data Interface (FDDI) discussed in Volume II also uses a token passing technology.

8.7.1 Token Ring (IEEE 802.5)

The basic operation of this MAC protocol is simple. When there is no traffic on the ring, a 3-byte token, illustrated in Fig. 8.11, circulates endlessly, waiting for a station to seize it by setting the T-bit or token bit the second byte to 1. Ignoring the priority mechanism which the token ring protocol allows for, the algorithm

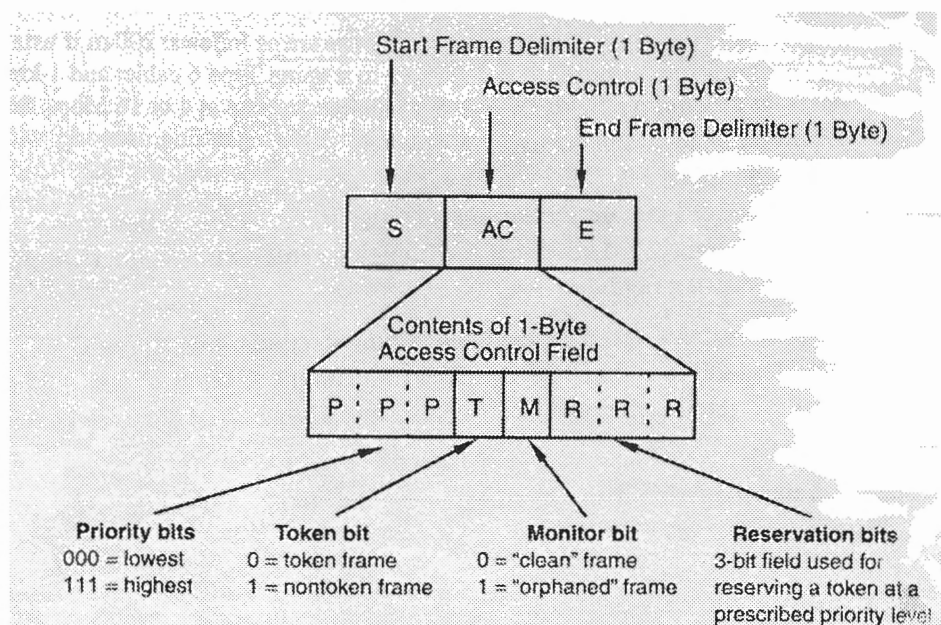


Figure 8.11: Token-Ring frame format

for accessing the medium is shown in the flowchart in Fig. 8.12. If $T = 0$ and the station has data to send, the token is seized by setting $T = 1$ which converts the first two bytes into the start-of-frame sequence. The station then outputs the rest of the data frame shown in the lower part in Fig. 8.13. Note that a station has to *delay* the token by one bit duration. In other words, it cannot pass the token on until it has decided whether

the bit currently arriving⁴ is a binary *one* or *zero* and it takes one bit interval for it to determine this. Under

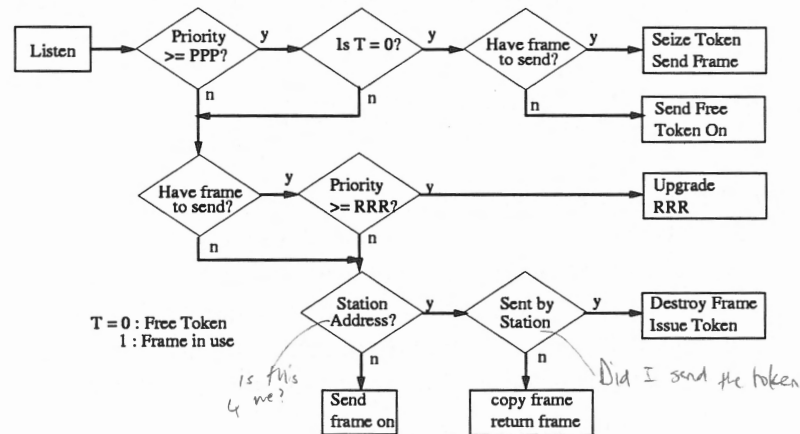


Figure 8.12: Token-Ring medium access algorithm

normal conditions, the first bit of the frame will circulate around the ring and return to the sending station before the full frame has been transmitted. Only a very long ring will be able to hold even a short frame. Consequently, the transmitting station must drain the ring while it continues to transmit by removing the bits which have completed the trip around the ring as they return to the sender. If the address is neither that of the destination or source they are simply passed on.

A station may hold the token for the *token-holding time*, which is generally 10 milliseconds. If there is enough time to send more frames after the first has been sent, these may also be sent. After either all frames have been transmitted or the transmission of another frame would exceed the token-holding time, the station regenerates the 3-byte token frame and inserts it into the ring.

The *starting delimiter* and *ending delimiter* fields of Fig. 8.13 mark the starting and ending of the frame. Each contains invalid differential Manchester patterns in order to distinguish them from data bytes. The *access control* byte contains the token bit, the *monitor bit*, *priority bits* and *reservation bits* (described later). Following the Access Control byte are the *destination* and *source address* fields. Next comes the data, which may be as long as necessary, provided that the frame can still be transmitted within the token-holding time. Next comes the *checksum* field.

The very last byte, is the *frame status* byte. This contains, amongst others, bits A and C. When a frame arrives at the interface of the destination station, the interface turns on the A bit as it passes through. If the interface copies the frame to the station, then the C bit is also turned on. (A station might fail to copy a frame due to lack of buffer space etc.)

When the sending station drains the frame from the ring, it examines the A and C bits. Four combinations are possible as also illustrated in Fig. 8.13.

A = 0 and C = 0: destination not present or not powered up.

A = 1 and C = 0: destination present but frame not accepted.

A = 0 and C = 1: destination not present and frame accepted in error.

A = 1 and C = 1: destination present and frame accepted.

⁴Remember we pointed out earlier that at high transmission speeds the leading edge of a bit will arrive back at the transmitter long before the trailing edge left.

Every token has to be delayed one bit time at every station.
 - need to check if it is a token.

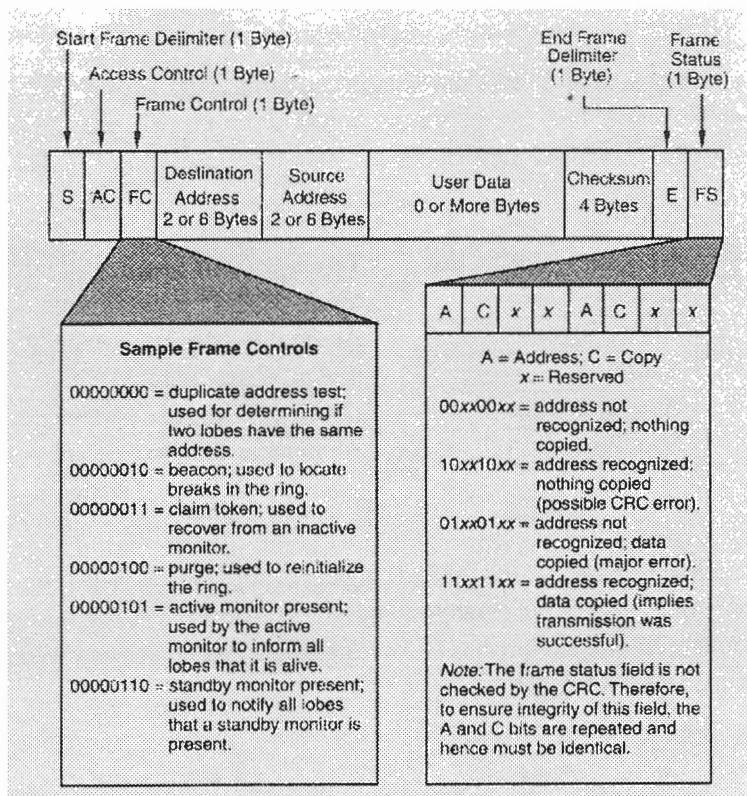


Figure 8.13: Token-Ring frame format and content

This arrangement provides for automatic acknowledgements for each frame. If a frame is rejected but the station is present, the sender can decide whether to resend the frame in a little while or not. The A and C bits are present twice in the *frame status* byte in order to increase reliability (since they are not covered by the checksum).

The *ending delimiter* contains an E byte, which is set if any interface detects an error. It also contains a bit that can be used to mark the last frame in a logical sequence, similar in purpose to an end-of-file bit.

The IEEE 802.5 protocol has an elaborate scheme for handling multiple priority frames illustrated in principle in Fig. 8.12. The 3-byte token frame has a field indicating the priority of the token. When a station wants to transmit a frame with priority *n*, it must wait for a token whose priority is less than or equal to *n*. Furthermore, when a data frame goes by, a station may attempt to reserve the next token by writing the priority of the frame it wants to send in the data frame's *reservation bits*. However, if a higher priority has already been reserved there, the station may not make a reservation. When the current frame has finished, the next token is generated at the priority that has been reserved.

It is easy to see that this mechanism works like a ratchet - in that it always jacks the priority higher and higher. To eliminate this problem, the protocol contains some complex rules. The basic idea is that the station raising the priority is also responsible for lowering it again when it is done.

8.7.2 Ring Maintenance

Each ring has a *monitor station* that oversees the ring. A contention protocol ensures that a new monitor is quickly chosen if the current monitor goes down (every station has the capacity to become monitor). It is

the monitor alone that is responsible for ensuring the correct operation of the ring.

When the ring comes up or any station notices that there is no monitor, it can send a *CLAIM TOKEN* control frame. If this frame makes its way back to the sender before any other *CLAIM TOKEN* frames, then the sender becomes the new monitor. The token ring control frames are shown in Fig. 8.11 and Table 8.2.

CONTROL FIELD	NAME	MEANING
00000000	Duplicate address test	Test if two stations have the same address
00000010	Beacon	Used to locate breaks in the ring
00000011	Claim Token	Attempt to become monitor
00000100	Purge	Re-initialize the ring
00000101	Active monitor present	Issued periodically by the monitor
00000110	Standby monitor present	Announces the presence of potential montitors

Table 8.2: Token ring control frames.

The monitor's duties include ensuring that the token is not lost, taking action when the ring breaks, cleaning the ring up when garbled frames appear, and watching out for orphan frames. An orphan frame occurs when a station transmits a complete frame, but crashes before the frame can be drained from the ring. This frame will circulate forever unless something is done.

The monitor handles these in the following way:

Lost tokens To check for lost tokens, the monitor has a timer which is set to the longest possible tokenless interval. If this timer goes off, the monitor drains the ring and issues a new token.

Garbled frames Garbled frames are detected by their invalid formats or their invalid checksums. The monitor drains the ring and issues a new token.

Orphans The monitor detects orphans by setting the *monitor* bit in the *Access control* byte whenever a frame passes through. If an incoming frame has this bit set, then it means that the frame has gone past the monitor twice without being removed - so the monitor drains it and a new token is issued.

Another monitor function concerns the length of the ring. If the token is 24 bits long, then the ring must be long enough to hold 24 bits. If the one length delays per station plus the delay due to the cable length is less than 24 bits, then the monitor inserts extra delay bits so that the token can circulate.

One function not handled by the monitor is detecting breaks in the ring. If a station notices that either of it's neighbours appears to be dead, a *BEACON* frame is transmitted, giving the address of the presumably dead station. When the beacon has propagated around as far as it can, it is then possible to see how many stations are down, and to delete them from the ring using the bypass relays in the wire centre - all without human intervention.

8.7.3 Token Ring Performance

A major issue in the design and analysis of ring networks is the "physical length" of a bit. If the data rate of the ring is R Mbps, a bit is emitted every $1/R$ microseconds. Bits are transmitted serially, just as in the case of Ethernet. With a typical signal propagation speed of 200 meters per microsecond, each bit occupies $200/R$ meters on the ring. This means, for example, that a ring whose circumference is 1000 meters can hold only 5 bits simultaneously.

Let us now briefly analyze the performance of a token ring. Assume that frames are generated according to a Poisson process (interarrival times have a negative exponential distribution), and that when a station receives permission to send, it empties itself of all queued frames, with the mean queue length being q frames at each station. The mean arrival rate of all N stations combined is λ so that, assuming identical behaviour at each of the stations, each station contributes λ/N to the traffic. Call the service rate (the number of frames/sec that a station can transmit) μ .

The mean time it takes for a bit to go all the way around an idle ring, or *walk time*, consisting of both the one bit-per-station delays and the mean *signal propagation delay*, plays a key role in the mean delay. Denote the walk time by ω . The quantity we intend to compute is the *scan time* s , the mean interval between token arrivals at a given station. *ie the time it would take a token to return to its sending station*

The scan time has two parts, the *walk time* w , and the time required to service each of the Nq requests queued up for service, each of which requires $1/\mu$ seconds to be transmitted. Algebraically,

$$(8.8) \quad s = \omega + Nq/\mu$$

The mean queue length q is easy to derive, since it is just the number of requests that pile up during an interval of length s when the arrival rate is λ/N , namely $q = \lambda s/N$. Substituting into the above equation we get

$$(8.9) \quad s = \omega + \lambda s/\mu$$

Setting $\rho = \lambda/\mu$ and solving for s we have

$$(8.10) \quad s = \omega/(1 - \rho).$$

The channel-acquisition delay is about half the scan time, so we now have one of the basic performance parameters. Notice that the delay is always proportional to the walk time, both for low and high load. Also note that ρ represents the utilization of the entire ring, not the utilization of a single station, which is ρ/N .

The other key performance parameter is the channel efficiency under heavy load. The only overhead is the walk time between stations. If every station has data to send, this overhead is ω/N , compared to the transmission time per station, Nq/μ . Using these times the channel efficiency can be found.

Although the token ring and Ethernet differ in many details, they also have many things in common. Since they use similar technologies, they can operate at similar data rates. In both cases stations must first acquire the channel before transmitting, with channel acquisition done by waiting for the token in one case and by contention in the other. Both systems have a channel efficiency determined by this wait time and transmission time. As the data rate is dramatically increased, say by using a fibre-optics medium, both networks' channel efficiency drops towards zero, because the channel-acquisition time in both cases depends on signal propagation speed, and this is not affected by increasing the data rate. Because the propagation time enters into the channel-acquisition time, both networks suffer from increased cable length, Ethernet's due to increased contention slot size and token rings due to increased walk time.

8.7.4 Token Passing Bus (IEEE 802.4)

As the name indicates, the token bus network is a bus network that uses a token-passing MAC protocol. IEEE 802.4 specifies a standard for the physical layer and the MAC sublayer of token bus networks.

A typical layout of a token bus network is shown in Fig. 8.14. The figure shows four nodes attached to a common coaxial cable. The nodes use a token-passing mechanism to regulate the access to the cable and

to avoid collisions. The figure illustrates one example of the sequence of events. Initially, node A holds the token and transmits a frame of data. The destination of that frame is any other node on the network. After having transmitted its data frame, node A transmits a token with destination address B. A token is a small frame identified as a token by the value of its access control field, as in an IEEE 802.5 token ring. All the nodes on the network see the token and recognise that it is for node B. Node B transmits its data frame and then the token with destination address C. Node C then transmits its data frame and the token with destination address D. Node D then transmits its data frame and followed by the token with destination address A. The transmissions continue in this cyclic order. This token-passing protocol is the *Token Bus MAC protocol*.

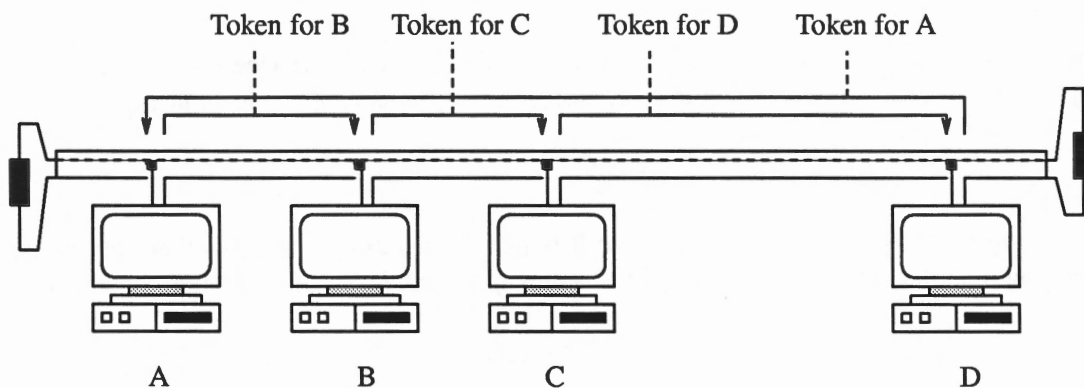


Figure 8.14: Token bus network

How does the network manager add or remove a node on a token bus network? This question does not arise on an Ethernet since there is no fixed order for the stations to transmit and the manager can plug a node in or take it out without having to inform the other nodes. The problem is also different on a token ring network since the connections are active and the ring stops operating when a node is disconnected. The nodes in a token bus network are either active or passive. A passive node is not participating in the communications and can be viewed as disconnected from the bus. In the example shown in Fig. 8.14, the successor of node A is B and the predecessor of node A is D. A node needs to know its successor to be able to send it the token.

By inserting or removing a node, the manager modifies the successor of a node. There is an automatic procedure for doing such modifications. Consider the network shown in Fig. 8.14 and assume that we want to insert a station E between stations C and D. Periodically, and only when it has the token, each node sends a special packet called a SAS (for *solicit a successor*). Any node that wishes to become active can reply just after the SAS has been transmitted. If no node replies, then the nodes resume the normal operation of the token bus MAC protocol. If one node replies to the SAS, then the other nodes modify their predecessor and their successor as needed to insert that new node. For instance say node C sends the SAS and node E replies *I want to join*.

The SAS sent by C indicates that C is its sender and that the successor of C is D. The reply from E indicates the address of E. When it sees the SAS from C, node E notes that its predecessor is now C and that the successor of E is D. When it gets the reply from E, node C replaces its successor D with E. If more than one node replies to the SAS, then the transmissions of these nodes collide. The collision is resolved as in CSMA-CD: the nodes whose transmissions collide repeat their transmissions with random delays, until one node is successful and is then inserted in the network.

When a node wants to be removed from the token bus, it waits until it gets the token. It then transmits a special message saying *I want out*. The other nodes then update their successor and predecessor information.

For instance, if node B on the network of our example sends the *I want out* message, that message includes the predecessor A and the successor C of B. When node A sees that message, it learns that its successor is no longer B but is instead C (B's successor). When C sees that its predecessor left the ring, it knows that its new predecessor is now B's old predecessor A.

When the network is started, or when the token is lost, the nodes detect the absence of activity. One node then sends an *I claim the token* packet. That node then sends the token to its successor if the lack of activity was due to a lost token. The node sends an SAS if this is the first transmission after starting the network. Collisions are handled as explained before.

The network deals with the failure of a node as follows: If node A sends the token to node B and node B does not repond, node A transmits a special message *Who follows B*. The nodes use the answer to that message to drop node B by updating their predecessor and successor.

8.8 Wireless LAN (IEEE 802.11)

As the number of portable and mobile computing and communication devices grows, so does the demand for connecting them in a convenient way. In order to achieve true mobility, portable devices need to use connectionless radio or infrared signals for communication. A system of devices that communicate by radio can be regarded as being connected by a Wireless LAN. These LANs obviously have somewhat different properties than conventional LANs and require their own MAC protocols as we will discuss next.

The IEEE 802.11 specification for Wireless LANs, approved in 1997, is based on a protocol designed in 1990, called *Multiple Access with Collision Avoidance (MACA)* which operate in the unregulated 2.4GHz to 2.4835 GHz frequency band with a bandwidth between 1 MBps and 2 MBps. The principle is that a station wishing to transmit, declares its intention by sending a short 30 byte *Request To Send (RTS)* frame. All stations within range will detect the RTS and refrain from transmitting anything themselves for the duration (indicated in the RTS) of the upcoming longer, data frame transmission.

Consider the diagram in Fig. 8.15 and assume station A want to transmit to B. A starts by sending an RTS as shown in Fig. 8.15 (a). The RTS of 30 bytes contains, *inter alia*, the length of the data frame to follow and the address of the destination station. B detects its own address and replies with a *Clear To Send (CTS)* signal, as shown in Fig. 8.15 (b) which again contains the length of the intended dataframe for reasons that will become clear. Upon receipt of the CTS frame, A will start transmitting its data frame. In the figure, station C is within range of A and thus hears the RTS from A, but not the CTS from B. Depending on its range (not shown) C is free to transmit while the data frame from A is being sent, provided it causes no interference. Station D, in turn, is within range of B (unshaded circle) but not station A (shaded circle) and therefore hears the CTS but not the RTS. Hearing the CTS and the length of the requested data transmission, it defers from sending anything until that frame should finish being transmitted. Station E hears both control messages and therefore remains silent until the end of (data) transmission from A.

Despite the algorithm, collisions do occur such as when stations B and C simultaneously decide to transmit an RTS. The transmissions will collide and be lost. In the event of a collision, the protocol behaves like CSMA/CD. In other words, an unsuccessful transmitter, i.e., one that does not get a CTS reply in the expected time interval, waits a random length of time using the same binary exponential backoff algorithm we already know, and tries again later until it succeeds.

Another class of radio network is cellular radio which is vastly popular for voice communication. Once the technology allows packet transmission over the wireless link, the Wireless Access Protocol (WAP) technology will doubtlessly have a major impact on wireless data communication. Cellular networks deserve a chapter of their own though which is what you will discover in Volume II.

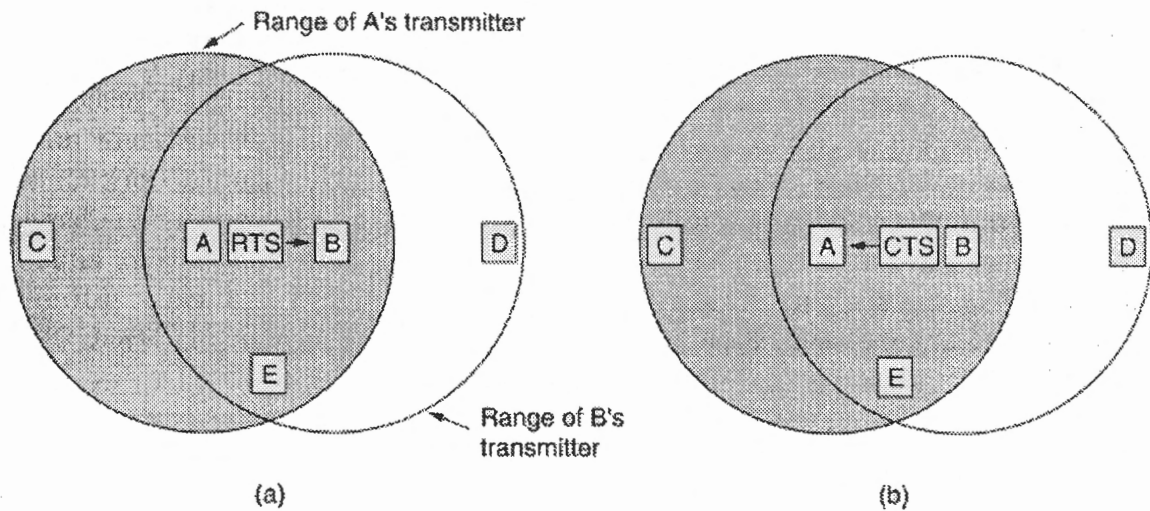


Figure 8.15: Wireless LAN communication

8.9 Network Interconnection

Many organisations have multiple LANs that they want to interconnect. The device used to effect interconnection depends on the highest layer in which there are difference between the protocols being inter-linked.

From the discussion of Ethernet, Token-bus and Token-ring LANs, it is obvious that two networks can differ greatly in access methods, frame sizes, frame formats, sequence numbering techniques, addressing conventions, and other items. In connecting two networks, the management information that originates in one network must be processable in the network in which it terminates and in any intermediate networks that it traverses. Also, the user's data must be understandable in the network in which it terminates. Differences between networks can be organised on the basis of the OSI model layer in which they occur:

1. Differences in the electrical and mechanical properties of the communication medium (e.g. cable vs. twisted pair, connectors, signaling methods, etc.) are differences in the implementation of the *physical* layers of the two networks.
2. Differences in the types of frame information and the significance of individual bit positions (e.g. synchronizing patterns and flags, control bits, sequence numbering, error control, flow control. etc.) are differences in the implementation of the *data link* layers.
3. Differences in segment sizes and routing techniques (e.g., centralized versus distributed) are differences in the implementation of the *network* layers.
4. Differences in message flow control and message segmentation are differences in the implementation of the *transport* layers.
5. Differences in line discipline, failure recovery strategies and communication management techniques are differences in the implementation of the *session* layers.
6. Differences in codes used and encryption and compression techniques employed are differences in the implementation of the *presentation* layers.

Depending upon these differences we can identify the following classes of interconnecting devices.

Repeater: if the differences are limited to the implementation of the Physical Layer, Repeaters, devices that link two connections and amplify and regenerate signals, are used.

Bridge: if the differences reach no higher than the implementation of the Data Link layers, Bridges, devices that link identical networks (i.e. Ethernet to Ethernet) at the Data Link layer, are used.

Router: if the differences reach no higher than the implementation of the Network layers, Routers are used. Routers modify segment sizes, route frames between users on non-identical, but the same type of network (i.e. Token Ring to Token Ring, WAN to WAN) and connect at the Network level.

Gateway: if the differences involve the implementation of layers above the Network layer, a Gateway is used. Gateways are devices that link networks that are not of the same type (i.e. Ethernet to Token Ring, or LANs to MANs or X.25 networks) and connect at whatever layer is required.

As a generic description, we may call all four of these devices *relays* – they connect two networks and move information from one to the other. Depending on and whether protocol conversion is being done, one can write a lot more about them. We shall leave the reader with these definitions only and refer him to many good references on the subject for the details of a particular instance.

8.10 Exercises

Exercise 8.1 *Define a Local Area Network.*

Exercise 8.2 *Compare the ring and bus topologies for reliability and performance.*

Exercise 8.3 *What is the difference between CSMA and CSMA/CD.*

Exercise 8.4 *Why is it necessary to have a one bit delay at each station in a token ring?*

Exercise 8.5 *At a transmission rate of $\lambda = 5$ Mbps and a propagation speed of $\mu = 200$ m/ μ second, how many meters of cable is the 1-bit delay in a token ring interface equivalent to? Explain your answer. Repeat the question for the case $\lambda = 100$ Mbps. What are the implications, if any, for a simple Stop and Wait protocol?*

Exercise 8.6 *A 1-km long, 10 Mbps CSMA/CD LAN (not 802.3) has a propagation speed of 200 m/ μ sec. Data frames are 256 bits long, including 32 bits of header, checksum and other overhead. The first bit slot after a successful transmission is reserved for the receiver to capture the channel to send a 32-bit acknowledgement frame. What is the effective data rate, excluding overhead, assuming that there are no collisions?*

Exercise 8.7 *A heavily loaded 1-Km long, 10 Mbps token ring has a propagation speed of 200 m/ μ sec. Fifty stations are uniformly spaced around the ring. Data frames are 256 bits, including 32 bits of overhead. Acknowledgements are piggybacked onto the data frames and are thus included as spare bits within the data frames and are effectively free. The token is 8 bits. Is the effective data rate of this ring higher or lower than the effective data rate of a 10 Mbps CSMA/CD network?*

Chapter 9

Wide Area Networks

9.1 Objectives of this Chapter

Wide area networks such as those which comprise the Internet have characteristics such as delay, distance and error properties which are different from ordinary voice networks or Local Networks connecting users in a close community. The implications of this are studied in this chapter.

1. Starting by distinguishing between Datagram (or *connectionless* and Virtual Circuit or *connection-oriented*) service, we discuss
 - Routing techniques.
 - Congestion control.
 - Flow control.
 - Deadlock prevention.

in the general case. We continue to discuss

2. Frame Relay networks, and
3. discuss the fundamentals of Asynchronous Transmission Mode (ATM) networks as a network protocol.
4. We explain X.25 as the first general Network Layer protocol, which have now probably been overtaken in use by
5. The Internet Protocol (IP) which is similar, but different in important ways.

9.2 Overview

The Data Link Control protocols of Chapter 7 concerned the reliable and controlled exchange of frames over unreliable lines between adjacent NODE's. As the name indicates, the network layer provides the primary interface between hosts and the Service Provider and it is at this level that the responsibility of the carrier or service provider towards its users generally ends.

Data communication networks are organised hierarchically. Every host has the illusion that it can converse directly with every other host via some kind of communication channel. In fact, the host is really communicating with its local NODE (or NODE), which in turn communicates with other NODE's, and *eventually* the destination host. The illusion that hosts can talk to one another is maintained by the network Service Provider or, if you prefer the term, the Internet. The actual paths that the messages between two hosts travel is transparent to the hosts. **Data link** errors are handled in a transparent manner by the Service Provider network; this is a service provided by the link layer to the network layer. Other errors, such as *lost and duplicate packets*, however, do occur. How these errors are dealt with by the network is a question of what services are provided by the Network layer to the Transport layer.

9.3 Datagram Service, Virtual Circuit Service

One of the most important design questions is the nature of the *service* provided to the *transport layer* by the *network layer*. Alternatively, what properties should the Service Provider network have when viewed by the hosts? To make this somewhat philosophical point more concrete, if a host sends a message to another host, should it be able to count on the message arriving properly? In other words, should the Service Provider make error detection and correction transparent to the hosts? Every service provider handles data link errors in a transparent way, but other errors can still occur: principally lost and duplicate packets due to faulty NODE hardware or software.

A second example of a function that the network layer might perform is making sure that packets arrive in the same order they were sent. In fact, there are a number of other possible options as well, but in practice, two conceptual models dominate. In the *Virtual Circuit model*, the network layer provides the transport layer with a perfect channel: no errors, all packets delivered in order, etc. just like a telephone call would for voice communication. In the *Datagram model*, the network layer simply accepts messages from the transport layer and attempts to deliver each one as an isolated unit (see Fig. 5.10 on page 67 much the same way that letters are delivered by the postal service – and as we well know, messages may arrive late, out of order, or not at all).

The analogy with a public telephone network versus the public postal service is clear. As in the case of a public telephone network, in a Virtual Circuit service, there is an explicit *setup procedure*, followed by a data *transfer phase* (talk) and then an explicit *shutdown procedure*. Once the circuit has been established, individual data items do not have to carry destination addresses, because the destination was specified during setup. With datagrams there is no setup. Each Datagram is carried independently from every other one and the network layer makes no attempt to do error recovery. It should be obvious that as far as the Service Provider is concerned, a Datagram service is the more primitive of the two services.

Virtual circuits have the advantage that the service user does not have to go to the trouble of *ordering* the packets since they arrive in order. However, this advantage turns into a disadvantage if a particular user process does not care about the ordering. For transaction oriented systems, such as point-of-sale credit verification, the cost of setting up and later clearing the Virtual Circuit can easily dwarf the cost of sending the single message.

A similar situation holds with *error control*. With Virtual Circuit service the user need not worry about errors, but there are situations where automatic retransmission is undesirable. An obvious example is digitized real-time voice. It is far better for the listener to hear an incorrect packet (a fraction of a word garbled) than to hear a 2 seconds pause while waiting for timeout and retransmission. In this case, the user might consider the error control provided by the Service Provider "too good".

In other situations however, the user might consider the degree of error control provided by the Service Provider too weak. Banks, in particular, tend to get quite upset if 1 dollar becomes 4095 dollars due to a transmission error. Those applications requiring better accuracy than the Service Provider provides (essentially longer checksums) may end up superimposing their own error-detecting scheme on top of the Service Provider's by computing and transmitting checksums within the data itself. In this case it is clearly inefficient to have the Service Provider duplicate what the user insists upon doing for himself anyway.

In addition to providing automatic sequencing and error control, virtual circuits also provide *flow control*. As in the other cases, there may be users who can handle very high data rates and do not want the Service Provider to impede them. For example, if one of the hosts is a workstation collecting real time data at high speed, and another is a dedicated one whose job is to store the data as fast as they come in, datagrams may be able to provide substantially higher throughput than a compulsory flow control scheme based on a *sliding window*. This conclusion is especially true if the Service Provider forces a standard window size on all customers.

Yet another potential advantage of datagrams for the sophisticated user is the ability to do his own packet switching. A router might take packets from the Service Provider as they come and inject them into a local private network. If the private network uses datagrams internally, there is hardly any point in forcing them to come out of the public ones in sequence, only to be randomized milliseconds later anyway. Yet if the Service Provider network only offers virtual circuit service, there will surely be a performance penalty to be paid.

From these examples you can see that datagrams provide sophisticated users with great flexibility in handling *sophisticated* applications. The supporters of virtual circuits would argue that most network applications (e.g., remote login, file transfer) prefer sequential, error free, flow controlled communication channels. Indeed this is the case with the Internet which ordinarily provides a Virtual Circuit service (TCP) although Datagram service (UDP) is available.

Most existing user programs and operating systems understand Virtual Circuit service, which makes interfacing to them easier. Also, by putting the sequencing, error control, and flow control in the Service Provider, the code only has to be written once. If these functions are in the hosts, each host operating system has to be modified, resulting in much more work. Replacing bits and pieces of code in everybody's operating system is a task to be favourably compared only with Hercules cleaning out the Augean stables. If the Service Provider designers are computer network experts and the computer centres owning the hosts are relative newcomers to networking, the argument for having the Service Provider designers do as much of the work as possible becomes even stronger.

Another argument for putting as much of the data communication work as possible in the Service Provider is that the router of the nodes are specialised dedicated processors, and invariably better suited to that kind of work than the hosts.

Table 9.1 contains a comparison of a Virtual Circuit and Datagram service.

ISSUE	VIRTUAL CIRCUIT	DATAGRAM
Initial setup	Required	Impossible
Destination address	During setup only	Needed in every packet
Packet sequencing	Packets passed to the host in the order received by Service Provider	Packets passed to the host in any order
Error handling	Explicitly done in Service Provider	Explicitly done by host
Flow control	Provided by Service Provider	Not provided by Service Provider

Table 9.1: Summary of the major differences between Virtual Circuit and Datagram service.

9.4 Internal Structure of the Network

Given the two classes of service the Service Provider can provide to its customers, it is time to see how the service provider network or simply “network” works inside. Interestingly enough, the same two models, Virtual Circuit and Datagram, can be used for internal transport. When Virtual Circuits are used, a route is chosen from source NODE to destination NODE when the circuit is set up. That route is used for all traffic flowing over the Virtual Circuit until it is torn down.

If packets flowing over a given Virtual Circuit always take the same route through the Service Provider, each NODE must remember where to forward packets for each of the currently open virtual circuits passing through it. Every NODE must maintain a table with one entry per open Virtual Circuit. Virtual circuits not passing through NODE X are not entered in X’s table, of course. Each packet travelling through the Service Provider must contain a Virtual Circuit number field in its header, in addition to sequence numbers, checksums, and the like. When a packet arrives at a NODE, the NODE knows on which line it arrived and what the Virtual Circuit number is. Based on this information, the packet must be forwarded to the correct NODE.

When a user process informs its local NODE that it wants to talk to a process in another machine, the router at that NODE chooses a Virtual Circuit number not already in use on that machine for the conversation. Since each NODE chooses Virtual Circuit numbers independently, the same Virtual Circuit number is likely to be in use on two different paths through some intermediate NODE, leading to ambiguities.

Consider the Service Provider network of Fig. 9.1. Suppose that a process in at NODE A wants to talk to a process at NODE D. NODE A chooses Virtual Circuit 0. Let us assume that route ABCD is chosen. Simultaneously, a process in B decides it wants to talk to a process in D (not the same one as A). If there are no open Virtual Circuits starting in B at this point, NODE B will also choose Virtual Circuit 0. Further assume that route BCD is selected as the best one. After both virtual circuits have been set up, the process at A sends its first message to D, on Virtual Circuit 0. When the packet arrives at D, the poor NODE does not know which user process to give it to.

To solve this problem, whenever a NODE wants to create a new outbound Virtual circuit, it chooses the lowest circuit number not currently in use. The NODE (say X) does not forward this setup to the new NODE (say Y) along the route as is. Instead, X looks in its table to find all the circuit numbers currently being used for traffic to Y. It then chooses the lowest free number and substitutes that number for the one in the packet, overwriting the number chosen by the user. Similarly, NODE Y chooses the lowest circuit number free between it and the next NODE.

When this setup packet finally arrives at the destination, the NODE there chooses the lowest available inbound circuit number, overwrites the circuit number found in the packet with this, and passes it to the

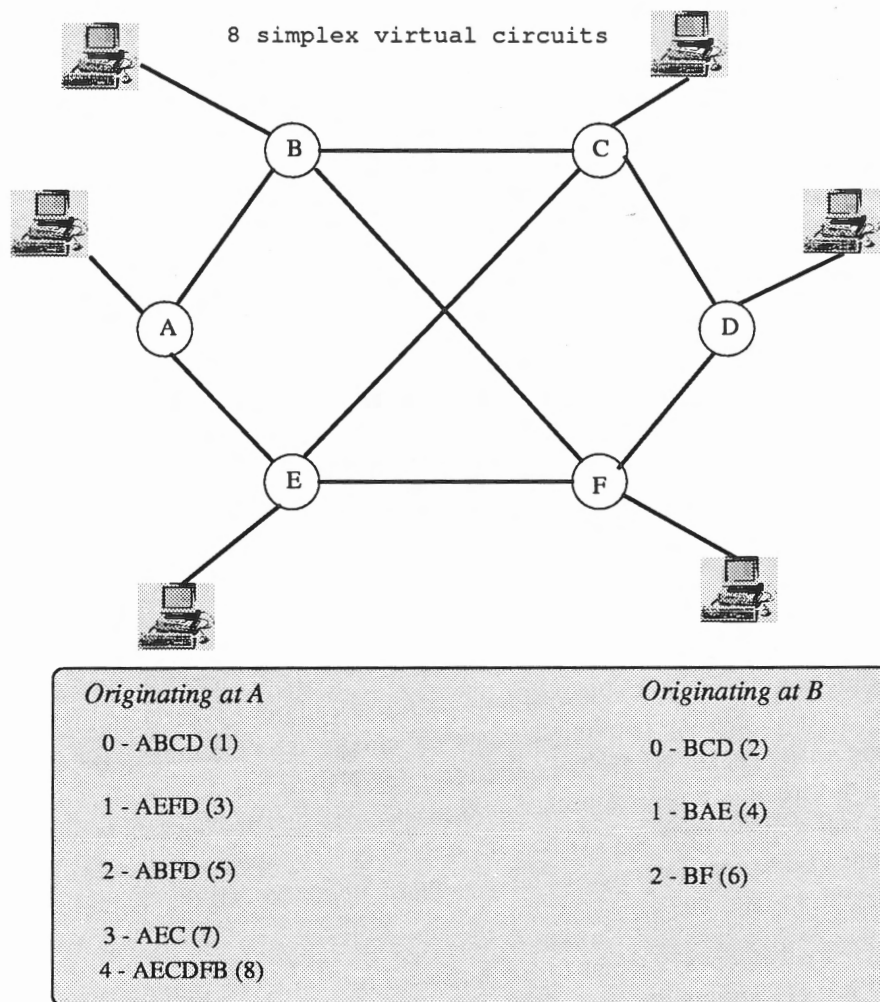


Figure 9.1: One directional Virtual Circuits showing the order on which they are set up

user. As long as the destination host always sees the same circuit number on all traffic arriving on a given Virtual Circuit, it does not matter that the source user is consistently using a different number.

Each entry consists of an incoming and an outgoing part. Each of the two parts has a NODE name (used to indicate a line) and a Virtual Circuit number. When a packet arrives, the table is searched on the left (incoming) part, using the arrival line and Virtual Circuit number found in the packet as the key. When a match is found, the outgoing part of the entry tells which Virtual Circuit number to insert into the packet and which NODE to send to it. H stands for the host, both on the incoming and outgoing sides.

Fig. 9.1 shows the above mentioned process of setting up the virtual circuits mentioned in the order shown in that figure. For example, consider a packet travelling from a user at NODE A to user at NODE B on Virtual Circuit (VC) 4 (i.e., route AECFB). When NODE A gets a packet from its user, with circuit 4, it searches its table, finding a match for H4 at entry 5 (the top entry is 0). The outgoing part of this entry is E3, which means replace the circuit 4 with circuit 3 and send it to NODE E. NODE E then gets a packet from A with circuit 3, so it searches for A3 and finds a match at the third entry. The packet now goes to C as circuit 1. The sequence of entries used is marked by the heavy line.

Because virtual circuits can be initiated from either end, a problem occurs when call setups are propagating in both directions at once along a chain of NODE. At some point they have arrived at adjacent NODEs. Each

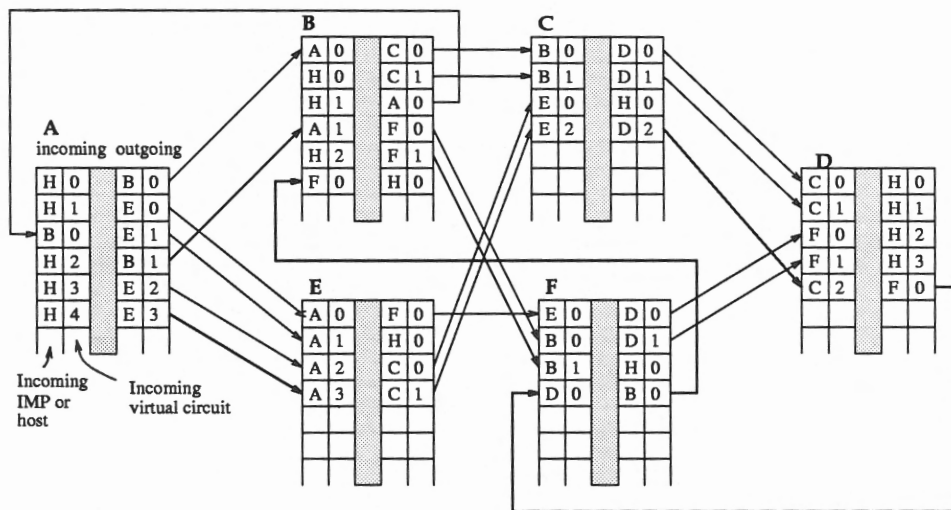


Figure 9.2: NODE tables corresponding to the Virtual Circuits set up Fig. 9.1

NODE must now pick a Virtual Circuit number to use for the (full-duplex) circuit it is trying to establish. If they have been programmed to choose the lowest number not already in use on the link between them, they will pick the same number, causing two unrelated virtual circuits over the same physical line to have the same number. When a data packet arrives later, the receiving NODE has no way of telling whether it is a forward packet on one circuit or a reverse packet on the other. We shall see later on how this conflict is resolved. If circuits are half-duplex, there is of course no ambiguity. One should realize that every process must be required to indicate when it is through using a Virtual Circuit, so that the Virtual Circuit can be purged from the NODE tables to recover the space.

So much for the use of virtual circuits internal to the Service Provider. The other possibility is to use Datagrams internally. When Datagrams are used, the NODEs do not have a table with one entry for each Virtual Circuit. Instead, they have a table telling which outgoing line to use for each possible destination NODE. These tables are also needed when Virtual Circuits are used internally to determine the route for a setup packet.

Each Datagram must contain the full destination address (the machine, and some indication of which process on that machine). When a packet comes in, the NODE looks up the outgoing line to use and sends it on its way. Nothing in the packet is modified. Also, the creation and destruction of Transport layer Virtual Circuits do not require any special work on the part of the NODEs.

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize the arguments both ways. Inside the Service Provider network the main trade-off between Virtual Circuits and Datagrams is between *NODE memory space* and *bandwidth*:

1. Virtual circuits allow packets to contain circuit numbers instead of full destination addresses. If the packets tend to be fairly short, a full destination address in every packet may represent a significant amount of overhead, and hence wasted bandwidth. The use of Virtual Circuits inside the Service Provider network becomes especially attractive when the users are online terminals, with only a few characters per packet. The price paid for using Virtual Circuits internal to the network is the table space within the NODEs or routers and the table lookup for every packet which arrives. Depending upon the relative cost of communication circuits versus NODE memory, one or the other may be cheaper.

For transaction processing systems such as is the case when the “users” are Automatic Teller Machines, the overhead required to set up and clear a Virtual Circuit may easily dwarf the use of the circuit, as mentioned above. If the majority of the traffic is expected to be of this kind, the use of Virtual Circuits inside the Service Provider network makes little sense.

Virtual circuits also have a vulnerability problem. If a NODE crashes and loses its memory, even if it comes back up a second later, all the Virtual Circuits passing through it will have to be aborted.

2. In contrast, if a Datagram NODE goes down, only those users whose packets were queued up in the NODE at the time will suffer, and maybe not even all those, depending upon whether they have already been acknowledged or not. The loss of a communication line is fatal to virtual circuits using it, but can be easily compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the Service Provider, since routes can be changed halfway through a conversation.

9.5 Routing Techniques

We will use the word “route” to mean the path that a packet follows as it traverses the Service Provider from its source NODE to its destination NODE. Since every NODE in a Wide Area Network routes packets along their way the term “router” is sometimes used instead of “NODE” when the context is that of routing only.

The *routing algorithm* is part of the Layer 3 software responsible for deciding which output line an incoming packet should be transmitted on. If the Service Provider network uses Datagrams internally, this decision must be made anew for every arriving data packet. However, if the Service Provider uses Virtual Circuits internally, routing decisions are made only when a new Virtual Circuit is being set up. Thereafter, data packets just follow the previously established route. The latter case is often called *session routing*, because a route remains in force for an entire session e.g., a login session at a terminal or a file transfer. In deciding upon a route that a packet should follow there must clearly be some objective. Minimizing mean packet delay is clearly one objective, but so is maximizing total network throughput. Furthermore, these two goals are also in conflict, since operating any queuing system near capacity implies a long queuing delay. As a compromise, many networks attempt to minimize the number of hops a packet must make, because reducing the number of hops tends to improve the delay and also reduce the amount of bandwidth consumed, which tends to improve the throughput as well.

Route selection may have to be based on many factors in different circumstances. These may pertain to carrier selection, use of an alternate route on the occasion of link failure, or load balancing among routes to reduce congestion. Consider the following: In certain situations, the lower cost of point-to-point satellite service may dictate use of that route for most traffic. During rain outages or eclipses of the satellite, on the other hand, terrestrial routes may be needed to ensure availability. A bulk transmission deferred to the early morning hours may best be routed via unused voice links for lowest cost. On the other hand, if the expected volume is higher than the voice links can handle, the use of a route via a nonswitched digital trunk may be the least cost. If the bulk transmissions are few in number, then a switched link may be less expensive.

Some traffic may merit a route at a premium cost in order to obtain very fast response time. In some of these illustrations, it is desirable that the origin NODE have a role in the route selection. In other situations, the route selections are entirely within one Service Provider’s domain, which appears as a link to the remainder of the network. These selections then may be based on workload distributions within that Service Provider network, the objective being to optimize the performance, for example, throughput or average response time, of the network.

Routing algorithms can be classified according to three characteristics: the *decision place*, the *information used* and the *decision time*.

1. **DECISION PLACE.** The place where the route is selected may be centralised or distributed.
 - (a) In a *centralised system*, the routing table for each NODE is determined at a central location and sent to each NODE; each NODE is given the route to every possible destination.
 - (b) In one type of *distributed decision*, the selection of the route to a given destination is based on the work loads; and the decision is made at each NODE along the route.
 - (c) In another type of *distribution decision*, such as explicit path routing, the selection of a route is made at the originating NODE for the message.
2. **INFORMATION SOURCE.** The route selection may be based on information concerning either link characteristics (for example, cost and availability) or link work loads. The source of this information may be
 - (a) The *originating NODE* of the message, if the selection of the route is by link cost or the matching of link characteristics to the message characteristics.
 - (b) A *NODE in the vicinity* of an outage, if the selection of the route is by link availability.
 - (c) *All NODEs* in the network, if the selection is based on work loads throughout the network.
 - (d) Those *NODEs immediately adjacent* to the NODE making the decision, if the selection is by work loads in the local environment.

Note that the current workload at any NODE can only be known precisely at that one NODE. Information about the workload at one NODE may be sent to another NODE, but that information is already somewhat out of date by the time it arrives. Global information in particular (being gathered from greater distances) is meaningful only in terms of averages over a period of time.
3. **DECISION TIME.** Selection of a new route to a given destination may be made frequently or infrequently. Occasions for route selection might be:
 - (a) The start of a packet.
 - (b) The start of a session.
 - (c) Major shifts in workload patterns.
 - (d) Link or NODE outages.
 - (e) Major shifts in network configuration or use of carrier services.

When multiple routes exist between a pair of nodes, the preferred route may change from time to time due to changing load conditions. How rapidly one should change the preferred route depends on the load characteristics themselves and the economic benefits to be derived, as well as the cost and difficulties one can encounter with too rapid a re-optimization. A conservative approach is to consider a re-optimization of routes on a relatively long-term basis. That can involve the employment of a slowly adaptive and largely heuristic optimization. The sending NODE could periodically sample the time delays encountered in the several available routes and could modify the route in accordance with this long-term sampling. However, shorter-term optimization will sometimes be advantageous.

Based on these three characteristics, routing algorithms can be grouped into two major classes: *nonadaptive* and *adaptive*. Nonadaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology, whereas adaptive ones do. If an adaptive algorithm manages to adapt well to the traffic, it will naturally outperform an algorithm that is oblivious to what is going on in the network, but adapting well to the traffic is easier said than done.

9.6 Congestion Control

When too many packets are present in the Service Provider network or in part of that network, performance degrades. This situation is called congestion and is depicted in Fig. 9.3.

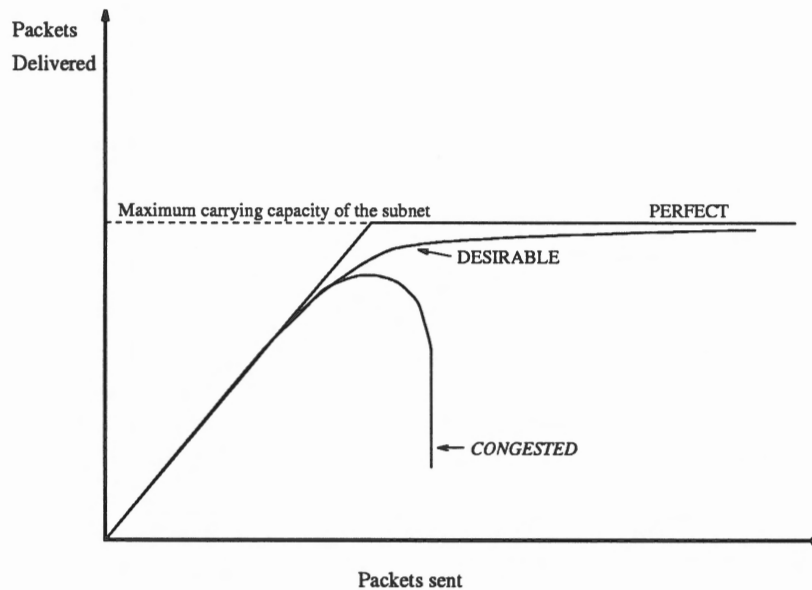


Figure 9.3: Congestion in the Service Provider network.

When the number of packets dumped into the Service Provider network by the users is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors), and the number delivered is proportional to the number sent. However, as traffic increases too far, the NODEs are no longer able to cope, and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely, and almost no packets are delivered.

Congestion can be brought about by several factors. If the NODEs are too slow to perform the various bookkeeping tasks required of them (queuing buffers, updating tables, etc.) queues can build up, even though there is excess line capacity. On the other hand, even if the router CPU is infinitely fast, queues will build up whenever the input traffic rate exceeds the capacity of the output lines. This can happen, for example, if three input lines are delivering packets at top speed, all of which need to be forwarded along the same output line. Either way, the problem boils down to not enough NODE buffers. Given an infinite supply of buffers, the NODE can always smooth over any temporary bottlenecks by just hanging onto all packets for as long as necessary. Of course, for stable operation, the hosts cannot indefinitely pump packets into the Service Provider network at a rate higher than that network can absorb.

Congestion tends to feed upon itself and become worse. If a NODE has no free buffers, it must ignore newly arriving packets. When a packet is discarded, the NODE that sent the packet will time out and retransmit it, perhaps ultimately many times. Since the sending NODE cannot discard the packet until it has been acknowledged, congestion at the receiver's end forces the sender to refrain from releasing a buffer it would normally have freed. In this manner congestion builds up, like cars in a traffic jam.

We will now examine five strategies for dealing with congestion:

1. Preallocation of resources to avoid congestion.

2. Allowing NODEs to discard packets at will.
3. Restricting the number of packets allowed in the Service Provider.
4. Using flow control.
5. Choking off input when congestion occurs.

9.6.1 Preallocation of Buffers

If Virtual Circuits are used inside the Service Provider network, it is possible to solve the congestion problem altogether, as follows. When a Virtual Circuit is set up, the Call Request packet winds its way through the Service Provider, making table entries as it goes. When it has arrived at the destination, the route to be followed by all subsequent data traffic has been determined and entries made in the routing tables of all the intermediate NODEs.

Normally, the Call Request packet does not reserve any buffer space in the intermediate NODEs, just table slots. However, a simple modification to the setup algorithm could have each Call Request packet reserve one or more data buffers as well. If a Call Request packet arrives at a NODE and all the buffers are already reserved, either another route must be found or a “busy signal” must be returned to the caller. Even if buffers are not reserved, some aspiring Virtual Circuits may have to be rerouted or rejected for lack of table space, so reserving buffers does not add any new problems that were not already there.

By permanently allocating buffers to each Virtual Circuit in each NODE, there will always be a place to store any incoming packet until it can be forwarded. First consider the case of a stop-and-wait NODE-NODE protocol. One buffer per Virtual Circuit per NODE is sufficient for simplex circuits, and one for each direction is sufficient for full-duplex circuits. When a packet arrives, the acknowledgement is not sent back to the sending NODE until the packet has been forwarded. In effect, an acknowledgement means that the receiver not only received the packet correctly, but also that it has a free buffer and is willing to accept another. If the NODE-NODE protocol allows multiple outstanding packets, each NODE will have to dedicate a full window’s worth of buffers to each Virtual Circuit to completely eliminate the possibility of congestion. These principles are partially illustrated in Fig. 9.4.

When each Virtual Circuit passing through each NODE has a sufficient amount of buffer space dedicated to it, packet switching becomes quite similar to circuit switching. In both cases an involved setup procedure is required. In both cases substantial resources are permanently allocated to specific connections, whether or not there is any traffic. In both cases congestion is impossible because all the resources needed to process the traffic have already been reserved. And in both cases there is a potentially inefficient use of resources, because resources not being used by the congestion to which they are allocated are nevertheless unavailable to anyone else.

Because dedicating a complete set of buffers to an idle Virtual Circuit is expensive, some Service Providers may use it only where low delay and high bandwidth are essential, for example on Virtual Circuits carrying digitized speech. For Virtual Circuits where low delay is not absolutely essential all the time, a reasonable strategy is to associate a timer with each buffer. If the buffer lays idle for too long, it is released, to be re-acquired when the next packet arrives. Of course, acquiring a buffer might take a while, so packets would have to be forwarded without dedicated buffers until the chain of buffers could be set up again.

9.6.2 Discarding Packets

Our second congestion control mechanism is just the opposite of the first one. Instead of reserving all the buffers in advance, nothing is reserved in advance. If a packet arrives and there is no place to put it, the

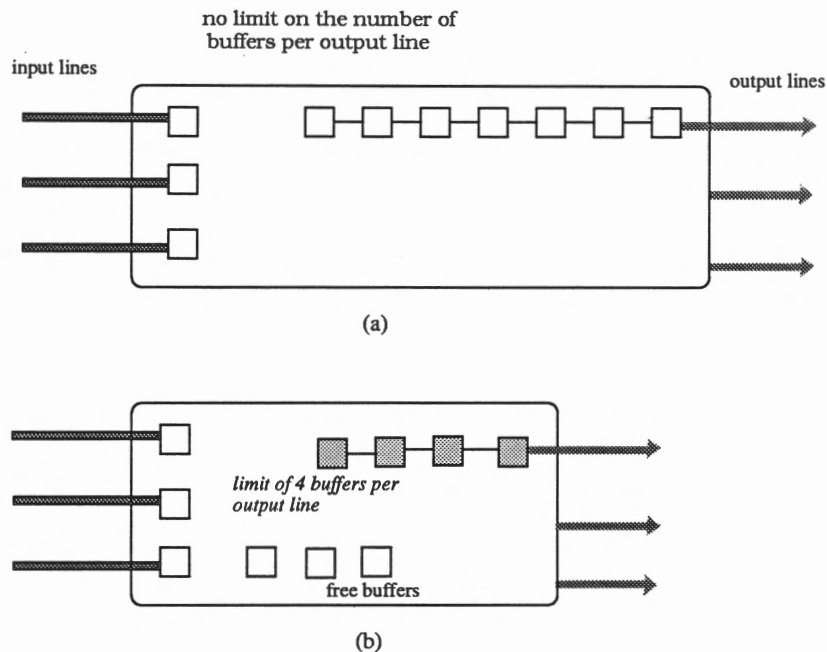


Figure 9.4: Congestion in a NODE can be reduced by putting an upper bound on the number of buffers queued on an output line

NODE simply discards it. If the Service Provider network offers Datagram service to the users, that is all there is to it: congestion is resolved by discarding packets at will. If the Service Provider network offers Virtual Circuit service, a copy of the packet must be kept somewhere so that it can be re-transmitted later. One possibility is for the NODE sending the discarded packet to keep timing out and re-transmitting the packet until it is received. Another possibility is for the sending NODE to give up after a certain number of tries, and require the source NODE to time out and start all over again.

Discarding packets at will can be carried too far. It is clearly stupid in the extreme to ignore an incoming packet containing an acknowledgement from a neighbouring NODE. That acknowledgement would allow the NODE to abandon a by-now-received packet and thus free up a buffer. However, if the NODE has no spare buffers, it cannot acquire any more incoming packets to see if they contain acknowledgements. The solution is to permanently reserve one buffer per input line to allow all incoming packets to be inspected. It is quite legitimate for a NODE to examine a newly arrived packet, make use of any piggybacked acknowledgement, and then discard the packet anyway. Alternatively, the bearer of good tidings could be rewarded by keeping it, using the just freed buffer as the new input buffer.

One way to minimize the amount of bandwidth wasted on the re-transmission of discarded packets is to systematically discard packets that have not yet traveled far and hence do not represent a large investment in resources.

9.6.3 Choke Packets

What is really needed for proper congestion control is a mechanism that is triggered only when the system is congested.

The following is one such mechanism: Each NODE monitors the percent utilization of each of its output lines. Associated with each line is a real variable, u , whose value, between 0 and 1, reflects the recent

utilization of that line. To maintain a good estimate of u , a sample of the instantaneous line utilization, f (either 0 or 1), can be made periodically and u updated according to

$$u = au + (1 - a)f$$

where the constant a determines how fast the NODE forgets recent history.

Whenever u moves above the threshold, the output line enters a “warning” state. Each newly arriving packet is checked to see if its output line is in warning state. If so, the NODE sends a *choke packet* back to the source NODE, giving it the destination found in the packet. The packet itself is tagged (a header bit is turned on) so that it will not generate any more choke packets later, and is forwarded in the usual way.

When the source NODE gets the choke packet, it is required to reduce the traffic to the specified destination by X percent. Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the NODE should ignore choke packets referring to that destination for a fixed time interval. After that period has expired, the NODE listens for more choke packets for another interval. If one arrives, the line is still congested, so the NODE reduces the flow still more and begins ignoring choke packets again. If no choke packets arrive during the listening period, the NODE may increase the flow again. The feedback implicit in this protocol should prevent congestion, yet not throttle any flow unless trouble occurs.

Several variations on this congestion control algorithm have been proposed. For one, the NODEs could maintain two critical levels. Above the first level, choke packets are sent back. Above the second, incoming traffic is just discarded, the theory being that the NODE has probably been warned already. Without extensive tables it is difficult for the NODE to know which NODEs have been warned recently about which destinations, and which NODEs have not.

Another variation is to use queue lengths instead of line utilization as the trigger signal. The same exponential weighting can be used with this metric as with u , of course. Yet another possibility is to have the NODEs propagate congestion information along with routing information, so that the trigger is not based on only one NODEs observations, but on the fact that somewhere along the path there is a bottleneck. By propagating congestion information around the Service Provider network, choke packets can be sent earlier before too many more packets are under way.

9.7 Flow control

There is some confusion in the literature between *flow control* and *congestion control*. Most authors use flow control to mean the mechanism by which a receiver throttles a sender to prevent data from arriving at a rate faster than the receiver can handle it. Stop-and-wait and the sliding window are two well-known flow control mechanisms. All flow control mechanisms either require the sender to stop sending at some point and wait for an explicit go-ahead message, or permit the receiver to simply discard messages at will with impunity.

Congestion control, in contrast, deals with the problem of more packets arriving at a *NODE* or router than there are buffers to store them all. Flow control is an end-to-end phenomenon, whereas congestion control deals with problems occurring at intermediate NODEs as well.

The reason some writers have confused the two topics is not hard to find. Some networks have attempted to use flow control mechanisms to eliminate congestion. Although flow control schemes can be used by the transport layer to keep one host from saturating another, and flow control schemes can be used to prevent

one NODE from saturating its neighbours, it is difficult to control the total amount of traffic in the network using end-to-end flow control rules.

Flow control cannot really solve congestion problems for a good reason: computer traffic is bursty. Most of the time an interactive user sits at his terminal scratching his head, but once in a while he may want to scan a large file. The potential peak traffic is vastly higher than the mean rate. Any flow control scheme which is adjusted so as to restrict each user to the mean rate will provide bad service when the user wants a burst of traffic. On the other hand, if the flow control limit is set high enough to permit the peak traffic to get through, it has little value as congestion control when several users simultaneously demand peak throughput.

Flow control can be used to limit the traffic between pairs of:

1. User processes, e.g., one outstanding message per Virtual Circuit.
2. Hosts, irrespective of the number of virtual circuits open.
3. Source and destination NODEs, without regard to hosts.

Another form of flow control is to limit the number of virtual circuits open.

9.8 Deadlock Prevention

The ultimate congestion is a *deadlock*, also called a *lockup*. The first NODE cannot proceed until the second NODE does something, and the second NODE cannot proceed because it is waiting for the first NODE to do something. Both NODEs have ground to a complete halt and will stay that way forever. Service Providers do not consider deadlock as a desirable property to have in their networks ...

The simplest lockup can happen with two NODEs. Suppose that NODE A has five buffers, all of which are queued for output to NODE B. Similarly, NODE B has five buffers, all of which are occupied by packets needing to go to NODE A. The situation is illustrated in Fig. 9.5(a). Neither NODE can accept incoming packets from the other. They are both stuck. This situation is called *direct store and forward lockup*.

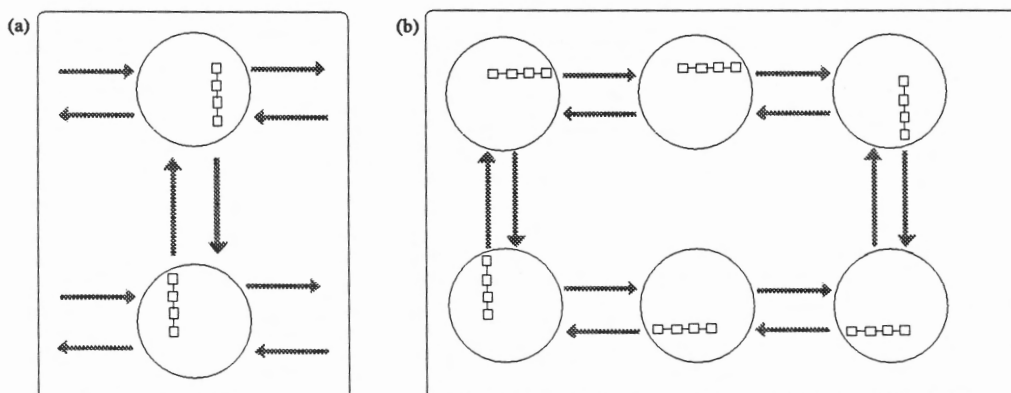


Figure 9.5: Store-and-forward lockup: (a) direct, (b) indirect.

The same thing can happen on a larger scale, as shown in Fig. 9.5(b). Each NODE is trying to send to a neighbour, but nobody has any buffers available to receive incoming packets. This situation is called *indirect store-and-forward lockup*. Note that when a NODE is locked up, all its lines are effectively blocked, including those not involved in the lockup.

Merlin and Schweitzer (1980) have presented a solution to the problem of store-and-forward lockup. In their scheme, a directed graph is constructed, with the buffers being the nodes of the graph. Arcs connect pairs of buffers in the same NODE or adjacent NODEs. The graph is constructed in such a way that if all packets move from buffer to buffer along the arcs of the graph, then no deadlocks can occur.

Store-and-forward lockups are not the only kind of deadlock that plague a Wide Area Network. Consider the situation of a NODE with 20 buffers and lines to 4 other NODEs, as shown in Fig. 9.6. Four of the buffers

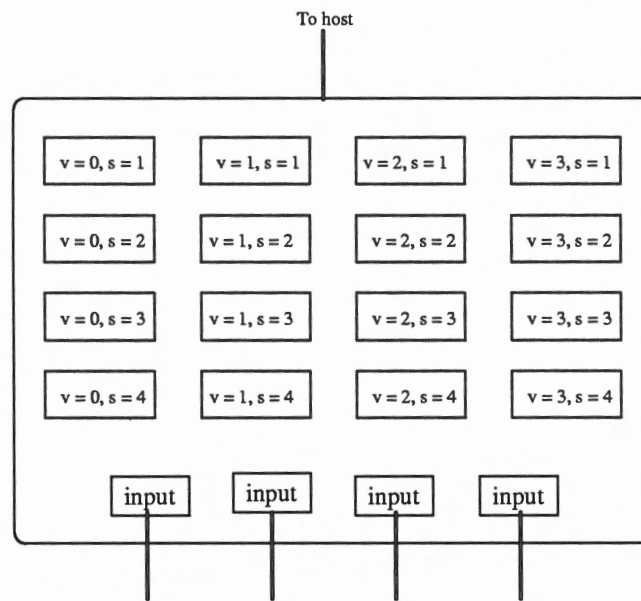


Figure 9.6: Message re-assembly lockup.

are dedicated to the 4 input lines, to help alleviate congestion. Assume that a sliding window protocol is being used, with window size 7. Further, assume that packets are accepted out of order by the NODE, but must be delivered to the host in order. At the time of the snapshot, 5 virtual circuits, 0, 1, 2, 3, and 4, are open between the host and other hosts.

Rather than dedicate a full window load of buffers to each open virtual circuit, buffers are simply assigned on a first come, first served basis. As soon as the next sequence number expected by the host becomes available, that packet is passed to the user (with each Virtual Circuit independent of all the others). However higher number packets within the window are buffered in the usual way.

In Fig. 9.6, v and s indicate the Virtual Circuit and sequence number of each packet, respectively. The host is waiting for sequence number 0 on all four virtual circuits, but none of them have arrived undamaged yet. Nevertheless, all the buffers are occupied.

If packets 0 should arrive, it would have to be discarded due to lack of buffer space. A result is no more packets can be passed to the host and no buffers freed. This deadlock is called *reassembly lockup*.

9.9 Frame Relay Networks

While Wide Area Networks (WANs) which are members of the Internet network community, would all have an explicit Layer 3 protocol, WANs which use techniques other than packet switching have become

very popular during the last decade. Two such techniques are Frame Relay and Asynchronous Transmission Mode (ATM) which we shall now discuss.

Frame Relay has its origins in the Integrated Services Data Network (ISDN) specification of the 1980's. ISDN is designed to connect the user with the ability to connect to many different types of network service. These include Circuit Switched data connection, Voice Switching services, Packet Switching services and access to Frame Relay services. The *Frame Relay service* in ISDN was designed to provide a very high speed Packet Switching style data service. This was to be used for the inter-connection of devices requiring high throughput for short durations. It was soon realized that the principles behind Frame Relay were applicable outside ISDN. Consequently, the protocol was developed for use in a stand-alone environment.

In a Frame Relay network, the data are divided into variable length frames, all of which include address information. The frames are forwarded to the Frame Relay network which attempts to deliver the frames to the correct destination. So far, this is the same as for a normal Packet switched network such as X.25. However, the difference is in the implementation. Normal Packet switching operates at Layer 3 of the OSI model while Frame Relay operates at Layer 2 and does not implement all the functions normally associated with the Data Link Layer (DLL). In order to see the advantage of Frame Relay over normal packet switching, we compare typical X.25 and Frame Relay frames.

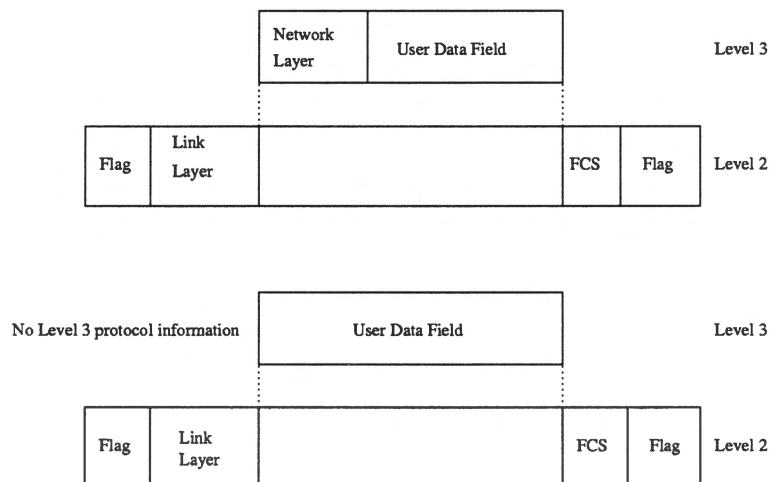
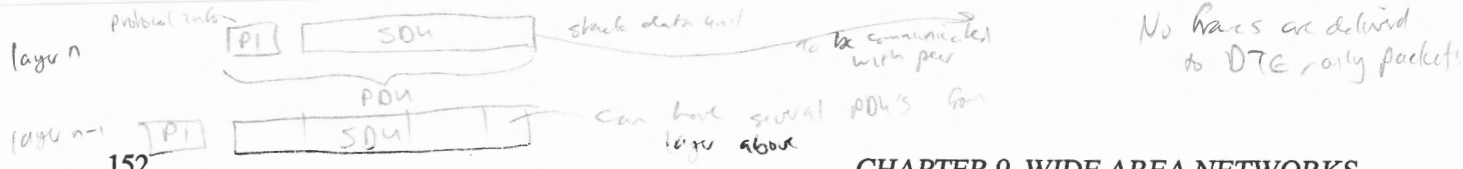


Figure 9.7: Packet Level (top) and Frame Relay (bottom) PDU formats

The PDU at the top of Fig. 9.7 reminds us of the structure as used by, for instance, the X.25 standard. There are five major components:

- **Flags:** These delineate the start and end of the packet. These are needed since the packets are not of fixed length.
- **Frame check sequence:** This typically contains a CRC to check whether there are any bit errors in the frame.
- **Link Layer component:** This is responsible for correct transmission and delivery of the packet. The Link Layer will attempt to recover from any errors. Also, this section contains flow control information.
- **Network Layer component:** This contains addressing information and ensures that a valid end-to-end connection is achieved. This contains the user's data but may also contain higher level protocol messages. In general, no more than 4096 bytes of user data are permitted in a single frame.



The detail required by the 3 level stack of the X.25 protocol is necessary for relatively unreliable links such as normally found. However, with the advent of new digital and fibre links whose bit errors are expected to be less than 10^{-10} , the amount of error control provided for by X.25 becomes unnecessary. Also, the user's applications are intelligent enough to detect errors and recover accordingly. In addition, the requirement for flow control at Layers 2 as well as 3 have the effect of decreasing the throughput of the network.

Frame relay was developed for very high speed packet switching style of networks. It works on basically two principles:

1. Packets (actually frames) which contain errors are simply discarded.
2. The end user is himself responsible for error recovery.

The main difference between Frame Relay and other forms of switching are that Frame Relay's functions are all implemented at the data link level. However, not all Packet Switching functions are implemented (those that are implemented are referred to as *core functions*). There are three main functions within Frame Relay:

1. Check for bit errors in frame. Discard frame if it contains an error.
2. Read address information and route to the appropriate output link.
3. Check that the Frame Relay switch is not congested. If it is, set the congestion notification bits or discard the frame.

In other words, unlike X.25, Frame Relay has no Layer 3 protocol i.e., it only has to access Layer 2 information instead of Layer 2 and Layer 3 information. In addition, the amount of processing that takes place at Layer 2 of Frame Relay is significantly less than that required at an interface such as X.25.

Fig. 9.8 demonstrates the difference between Packet Switching and Frame Relay. It is easy to see that the amount of processing done for Frame Relay is significantly less than Packet Switching and therefore has the potential for much greater throughput.

9.9.1 Circuit Connections

Frame relay was designed to be a simple protocol. It was initially defined as a permanent connection oriented protocol i.e., a connection between two users is permanently established and cannot be cleared down at any time. This mode of operation is referred to as a permanent virtual circuits (PVC) and is reasonably restrictive. The connection is established by the network administrator when the network goes up and is available until the network shuts down. There are moves to create an on-demand Frame Relay service (known as switched virtual circuits or SVC).

9.9.2 Committed Information Rates

One of the major advantages of Frame Relay is in handling bursts of data from the user. Since there is no flow control, users can send as much data into the network as required at any moment in time. However, this is a disadvantage to the network provider since there is no effective way of sizing the capacity of the backbone network. It is possible that all attached users could decide to flood the network with data at any particular moment in time.

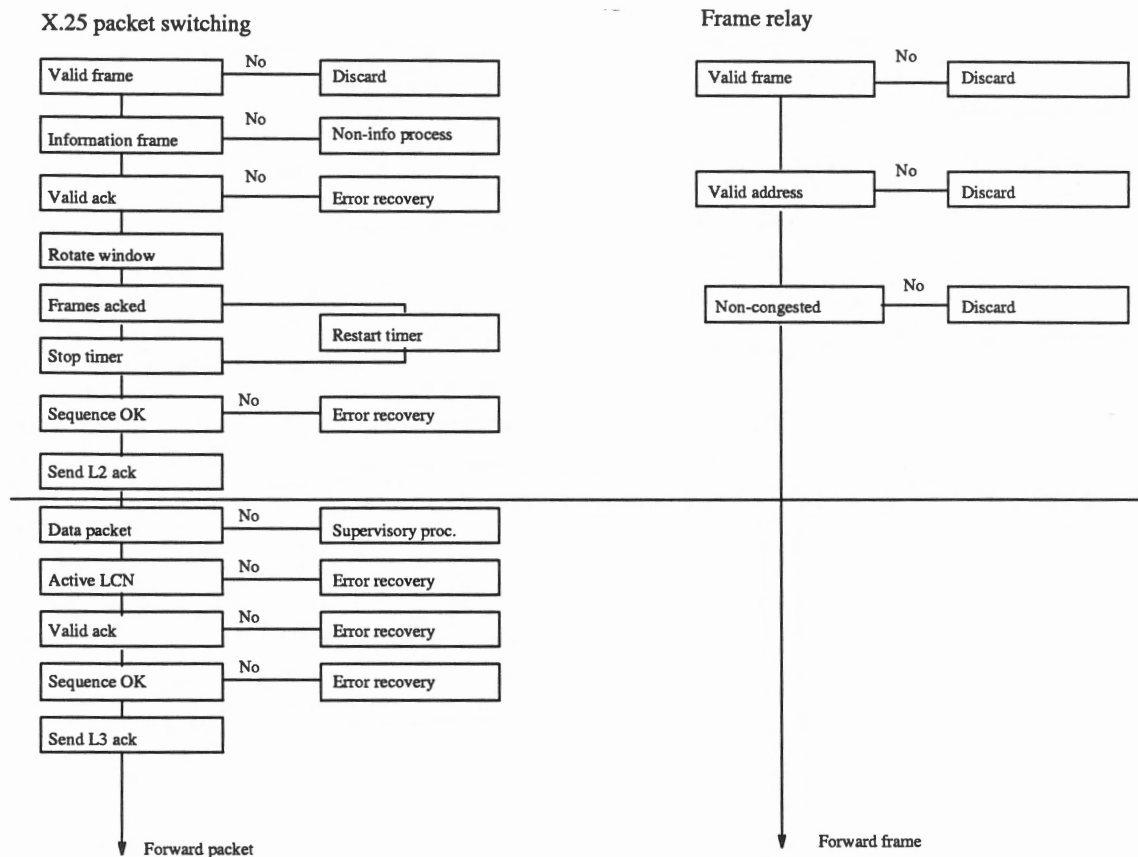


Figure 9.8: Packet and Frame Relay processing flow diagrams.

There are mechanisms to notify the user that there may be a problem if the user continues to send data into the network. However, there is no way to force the user to stop sending. To avoid the situation where a badly behaved user application continues to forward data in these conditions, the principle of *Committed Information Rates (CIR)* was developed.

A CIR is the average data rate which the user expects to pass data into the network which the network should be able to handle with few problems. Note that the CIR is unrelated to the actual bit rate of the physical link. For example, a user could have a physical connection of 2 Mbps but a CIR of only 64 kbps. This means that the average data rate is equivalent to 64 kbps but that bursts up to 2 Mbps would be possible.

It is quite possible that the user will want to exceed his CIR at times. One of the advantages of Frame Relay is that it is quite possible for the user to exceed his CIR, the network will attempt to accept that data if the bandwidth is available. However, if the network runs into any problems, it will simply discard frames.

The network distinguishes between data that has been committed to (via the CIR) and data that exceeds the CIR by the *discard eligibility bits*. The idea is that in the event of problems, these frames will be discarded first. The exact method of determining which frames are discard eligible will be vendor dependent, but Fig. 9.9 shows a possible scheme based upon the CIR. In a time period T, a certain number of frames will fall within the CIR. When a user sends more frames than the CIR, these extra frames are marked discard eligible.

CIR's are particularly relevant when used in the context of a public Frame Relay service. A public service could offer CIR's at different tariffs, but provide all users with the same physical connection. The public

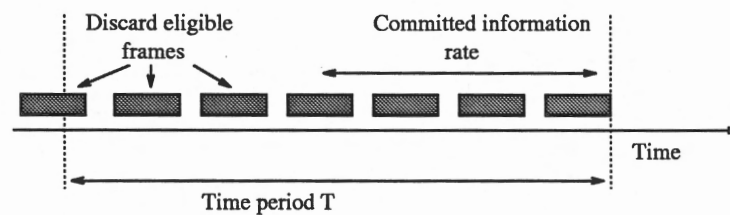


Figure 9.9: Illustrating Committed Information Rate (CIR) and discard eligibility of frames

service could then guarantee delivery of data that falls within the CIR with the ability for the user to exceed this, knowing the risks involved.

9.9.3 Local Management Interface (LMI)

Since Frame Relay is very simple, there are several omissions from the basic protocol specification that are considered essential by other packet switched protocols. These essential features are contained within the local management interface (LMI).

The LMI protocol is responsible for:

1. Ensuring that the link between the user and the network is active.
2. Notifying the addition and deletion of PVCs.
3. Delivering status messages regarding the availability of circuits.

Because Frame Relay does not handle this information, a separate Virtual Circuit is established between the user and the network over which this information is passed (this is known as *out of band* signaling).

The LMI operates as a polling protocol between the user and the network. A poll and acknowledgement are exchanged at regular intervals. An optional extension, the asynchronous update message can also be included. This asynchronous update message passes information about the status of a single PVC whenever a change occurs in its status. This extension enables the frame relay interface to be more responsive to changes in network conditions.

9.9.4 Frame Relay Applications

Frame Relay was designed to provide a high throughput, low transit delay networking interface, limited scope for error recovery and relies on the end user systems to recover from problems. These are also traditionally the characteristics of a Local Area Network (LAN) (especially Ethernet, see Chapter 8). Table 9.2 shows how LAN characteristics compare to the various WAN transport methods. Since Frame Relay has the same characteristics as a LAN, it's logical that Frame Relay is very successful in areas that require extending LANs over a wide area.

The true beauty of Frame Relay networking for the LAN user is when public Frame Relay services are used. The user could subscribe to a CIR which would be tarified at a certain level, but the LAN user knows that if the traffic momentarily exceeds this CIR, the public Frame Relay service will attempt to deliver the data (assuming the network is not congested). This is particularly important in the LAN area because of the potential bursts of data.

LAN CHARACTERISTIC	TDM	PACKET SWITCHING	FRAME RELAY
High speed interconnection	Possibly true	Possibly true	TRUE
Low transit delays	TRUE	Possibly true	TRUE
Poor error recovery	TRUE	FALSE	TRUE
Reliance on end user protocols	TRUE	FALSE	TRUE
Bandwidth on demand	FALSE	TRUE	TRUE

Table 9.2: LAN and WAN characteristics compared

9.10 The X.25 Packet Interface

X.25 is the ITU-T protocol standard which defines the format and meaning of the information exchanged across the DTE/NODE interface for the layer 1, 2 and 3 protocols as illustrated in Fig. 9.10. Since this interface separates the Service Provider's equipment (DCE in ITU-T terminology) from the user's equipment (DTE), it is important that the interface be very carefully defined and understood.¹ *Note that ITU-T lays*

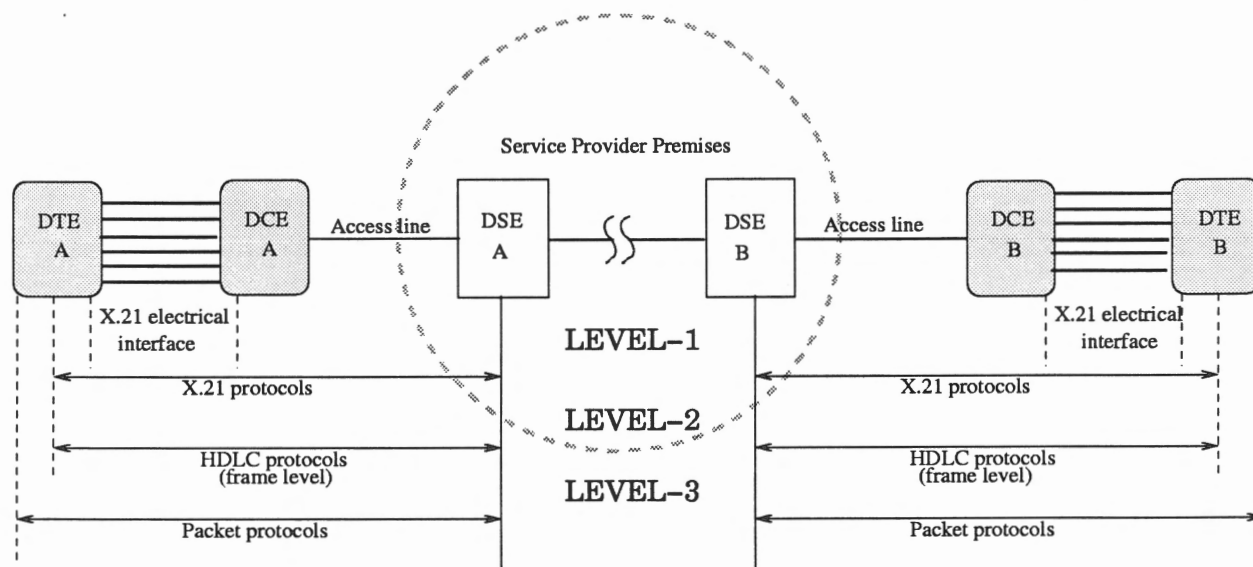


Figure 9.10: The DTE-DCE X.25 ITU-T interface standard for Wide Area packet switched networks.

down no standards as to the protocols used amongst NODE's internal to a packet switching network. Real digital or analog circuits extend, of course, from one NODE to another NODE – and between DTE's and NODE's on either end.

The X.25 protocol defines 2 layers (levels) of communication for switched digital networks. A third layer, the Network layer being added for packet switching networks. These are the following:

Level 1. For some analog and for some early digital networks, this is the EIA RS 232 or the equivalent ITU-T V.24 electrical interface. For newer circuit switched networks and for most new packet switching networks, this will be the ITU-T X.21 interface (although some packet switching networks will use X.21 BIS, which is basically the same as V.24). X.21 was discussed in Sec. 6.2.2 on page 72 and includes procedures for call establishment and clearing, which are network interface protocols.

¹ Although ITU-T distinguishes between the DCE (Data Circuit-terminating Equipment) and the Data-switching Equipment (DSE) as shown in the figure, the DCE and the DSE are usually indistinguishable from one another.

Level 2. The Data Link layer which ensures reliable communication between a DTE and a NODE, even though they may be connected by a noisy line. The protocols used are LAP and LAPB, which are very similar to SDLC discussed in Sec. 7.3 on page 101.

Level 3. A third level is needed in X.25 to support the user/NODE interface for packet switching networks. This third level is concerned with the information and meaning of the data field contained within each of the Level 2 frames. The packet layer provides for routing and Virtual Circuit management.

In packet-switched networks, *call establishment* and *clearing phases* for virtual circuits are evidenced by special packets formed at the third level. In addition, in packet-switched networks, there may be other *completely separate* call establishment/clearing phases at (the X.21) level 1 if the access to the packet-switched network is via a real digital (as opposed to analog) circuit.

Since the Layer 1 and Layer 2 protocols have already been described, the remainder of this section concerns the format and function of the Layer 3 packets. Each packet is enclosed in the data link control header and trailer as shown in Table 9.3.

FLAG	ADDRESS	CONTROL	FRAME	FRAME CHECK SEQUENCE (FCS)
------	---------	---------	-------	----------------------------

Table 9.3: X.25 packet format

When a NODE wants to communicate with another NODE, it must first set up a Virtual Circuit between them using a CALL REQUEST (CR) packet which has the format illustrated in Fig. 9.11.

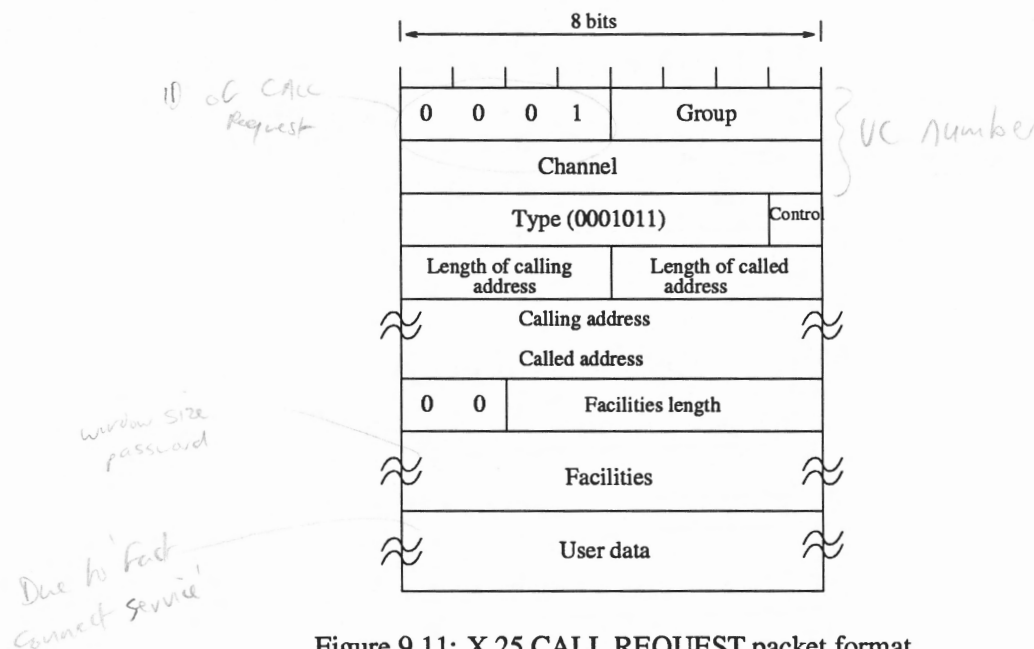


Figure 9.11: X.25 CALL REQUEST packet format.

The sequences in X.25 for the Virtual Circuit *call-establishment* phase, *data transfer* phase and *clearing* phase loosely resemble those of X.21, but use packets (rather than signals) for information exchange. A simplified illustration of this and the other phases is given in Fig. 9.12.

Remember, however, that the packets for Virtual Circuit establishment and clearing are formed at Level 3 and flow at Level 2, whereas the signals for call establishment clearing of a real access circuit appear at

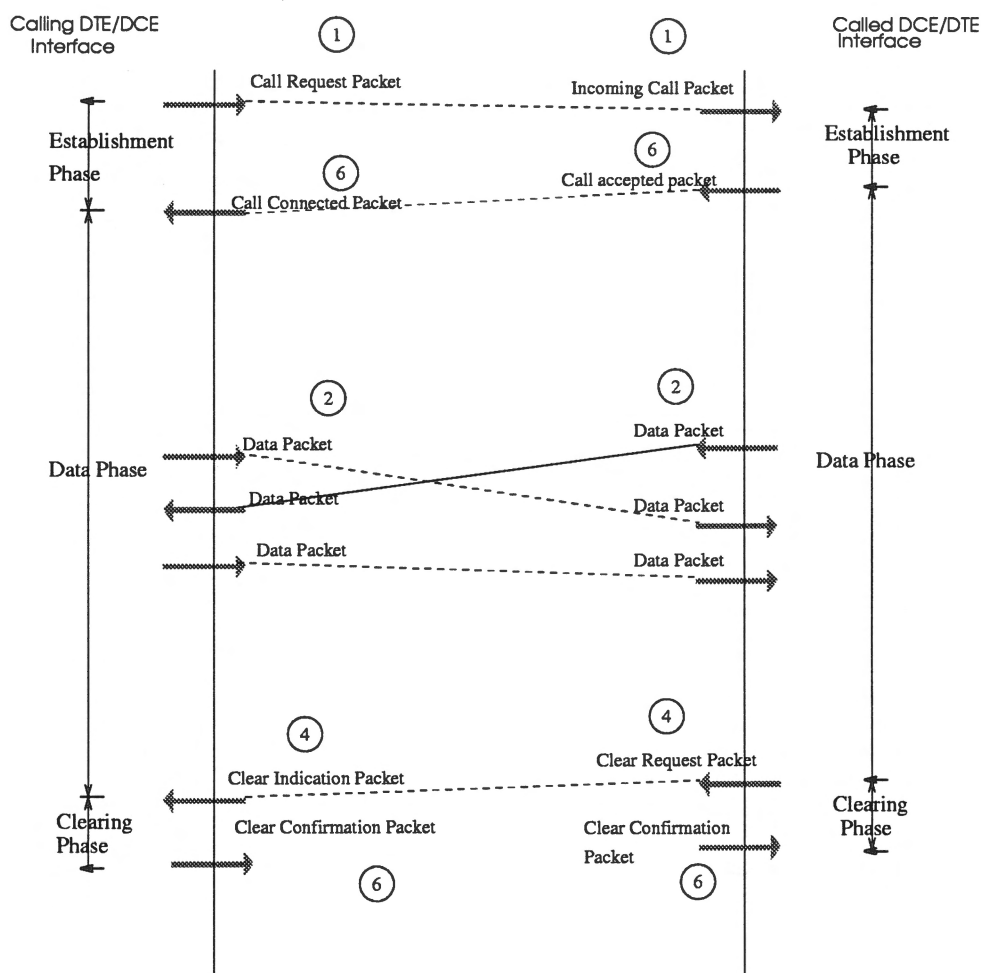


Figure 9.12: Sequence diagram of X.25 call establishment, data transfer and call clearing phases

Level 1 (X.21). Note also that the characters reporting the *call-progress* of a virtual circuit are placed in the additional information field of the *CLEAR REQUEST* or *RESET* packets whose format is illustrated in Fig. 9.13, whereas the characters of the call-progress signals of the real access circuits appear at Level 1.

The sequence shown in Fig. 9.12 can be described as follows:

1. The calling user/DTE builds a *CALL REQUEST* packet and passes it to its own NODE. This packet includes the called user/DTE address.
2. The Service Provider network delivers the packet to the destination NODE which indicates that there is an incoming call by sending to its DTE an *INCOMING CALL* packet (same format as the *CALL REQUEST* packet), using the logical channel that is in the ready state and has the lowest logical channel number.
3. If the called DTE wishes to accept the call, it sends a *CALL ACCEPTED* packet back. The latter specifies the same logical channel as that of the *INCOMING CALL* packet. The called DTE is now in the virtual call *data transfer state*.
4. The calling DTE also enters the data transfer state after it receives from its own NODE the *CALL CONNECTED* packet that specifies the same logical channel as the *CALL REQUEST* packet. Note

that the logical channel numbers at the two ends are usually different.

5. Alternatively, the destination NODE can advise the calling DTE that it is unable to connect a call by sending a *CLEAR INDICATION* packet. The calling DTE responds by dispatching a *CLEAR CONFIRMATION* packet and the calling DTE then returns to the ready state. The Additional Information field of the *CLEAR INDICATION* packet may be coded to indicate one of the following:

- number busy
- out of order
- remote procedure error
- number refuses reverse charging
- invalid call
- access barred
- local procedure error
- network congested
- not obtainable

item During the data transfer phase, traffic can flow in either direction at any time. Data, *INTERRUPT*, *FLOW CONTROL* and *RESET* packets may be exchanged during the data transfer state.

6. A DTE may request that the Virtual Call be cleared by sending a *CLEAR REQUEST* packet. However, the logical channel will not be returned to the ready state until the DTE is given a *CLEAR CONFIRMATION* packet by its own NODE.

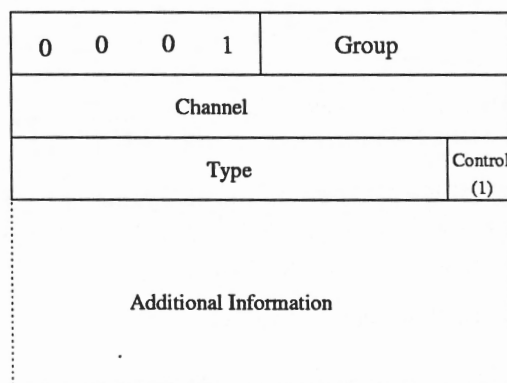


Figure 9.13: X.25 Control packet format

Note that the choice of circuit number on outgoing calls is determined by the calling DTE and on incoming calls by the called NODE. It could happen that both simultaneously choose the same circuit number, leading to a call collision. X.25 specifies that in the event of a call collision, the outgoing call is put through and the incoming call is cancelled. Many networks will attempt to put the incoming call through shortly thereafter using a different Virtual Circuit.

In addition to these Virtual Calls, X.25 also provides for *Permanent Virtual Circuits (PVC)*. These are analogous to leased lines in that they always connect two fixed DTEs and do not need to be set up.

Because X.25 makes a distinction between *CLEAR REQUEST* and *CLEAR CONFIRMATION*, there is the possibility of a clear collision, (i.e., both sides decide to terminate the Virtual Circuit simultaneously).

However, because it is always obvious what is happening, there is no ambiguity, and the Virtual Circuit can be cleared without delay.

All X.25 packets begin with a 3-byte (or 32-octet in ITU-T terminology) header as discussed next.

1. The *Group and Channel* fields together form a 12-bit Virtual Circuit number. Virtual Circuit 0 is reserved for future use, so in principle, a DTE may have up to 4095 Virtual Circuits open simultaneously. The Group and Channel fields individually have no particular significance.
2. The *Type* field in the CALL REQUEST packet, and in all other control packets, identifies the packet type. The *Control* bit is set to 1 in all control packets and to 0 in all data packets. By first inspecting this bit the DTE can tell whether a newly arrived packet contains data or control information.
3. The remaining fields of the CALL REQUEST packet illustrated in Fig. 9.11 are, to start with, the length of the *calling* and *called addresses*, respectively. Both addresses are encoded as decimal digits, 4 bits per digit.
4. The *Facilities* length field tells how many bytes worth of facilities field follows. The Facilities field itself is used to request special features for this Virtual Circuit. The specific features available may vary from network to network. One possible feature is *reverse charging (collect calls)*. This facility is especially important to organizations with thousands of remote terminals that initiate calls to a central computer. If all terminals always request reverse charging, the organization only gets one "network phone bill" instead of thousands of them. High-priority delivery is also a possibility. Yet another feature is a simplex, instead of a full-duplex, Virtual Circuit.

The caller can also specify a maximum packet length and a window size rather than using the defaults of 128 bytes and two packets, respectively. If the callee does not like the proposed maximum packet length or window size he may make a counterproposal in the facilities field of the CALL ACCEPTED packet. The counterproposal may only change the original one to bring it closer to the default values, not further away.

Some *facilities* must be selected when the customer becomes a network subscriber rather than on a call-by-call basis. These include *closed user groups* (no user can call outside the group, for security reasons), maximum window sizes smaller than seven (for terminals with limited buffer space), line speed (e.g., 2400 bps, 4800 bps, 9600 bps), and prohibition of outgoing calls or incoming calls (terminals place calls but do not accept them).

The USER data field allows the DTE to send up to 16 bytes of data together with the CALL REQUEST packet. The DTEs can decide for themselves what to do with this information. They might decide, for example, to use it to indicate which process in the DTE the caller wants to be connected with. Alternatively, it might contain a login password.

The Data or Information packet format is shown in Fig. 9.14. The Q bit indicates *Qualified* data and is used for *Quality of Service (QoS)* purposes.

The CONTROL field is always 0 for data packets. The PIGGYBACK and SEQUENCE fields indicate the lowest packet number expected by the sender (setting the lower edge of the flow control window) and the packet number of the packet being sent respectively, thus establishing flow control.

The sequence numbers are modulo 8 if the MODULO field is 01 and modulo 128 if MODULO is 10 (values 00 and 11 are illegal). If modulo 128 sequence number are used, the header is extended with an extra byte to accommodate the longer SEQUENCE and PIGGYBACK fields. The meaning of the PIGGYBACK field is determined by the setting of the D bit. If D = 0, a subsequent acknowledgement means only that

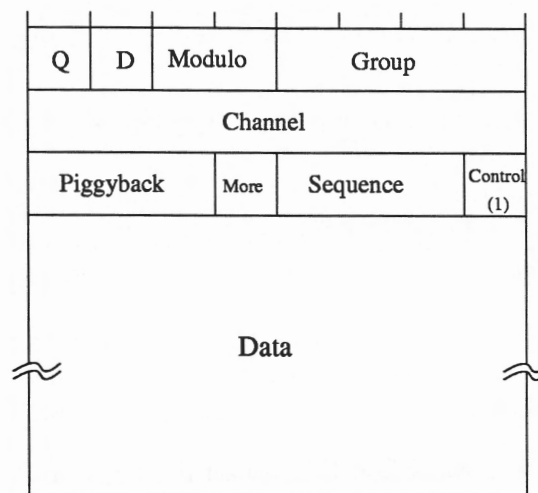


Figure 9.14: X.25 data packet format

the local NODE has received the packet, not that the remote user or DTE has received it². If $D = 1$, the acknowledgement is a true end-to-end acknowledgement, and means that the packet has been successfully delivered to the remote DTE.

The MORE field allows a DTE to indicate that a group of packets belong together. For example, to send a long message, each packet except the last one would have the MORE bit on. Only a full packet may have this bit set. The Service Provider is free to repackage data in different length packets if it needs to, but it will never combine data from different messages (as indicated by the MORE BIT) into one packet.

The standard says that all carriers are required to support a maximum packet length of 128 data bytes. However, it also allows carriers to provide optional maximum lengths of 16, 32, 64, 256, 512, and 1024 bytes. In addition, maximum packet length can be negotiated when a Virtual Circuit is set up. The point of maximum packet lengths longer than 128 is for efficiency. The point of maximum packet lengths shorter than 128 is to allow terminals with little buffer space to be protected against long incoming packets.

The original (1976) X.25 standard was more-or-less as we have described it so far (minus the D bit, DIAGNOSTIC packet, and the packet length and window size negotiation). However, there was considerable demand for a Datagram facility, in addition to the Virtual Circuit service. Both the United States and Japan made (conflicting) proposals for the architecture of the Datagram service. In the great tradition of international bureaucracies, ITU-T accepted both of them, making the protocol even more complicated than it already was. Both Datagram facilities are minor variations on the CALL REQUEST packet. In what is referred to as the true Datagram packet (U.S. proposal), the format is the same as the CALL REQUEST packet of Fig. 9.11, except that the third byte is replaced with a SEQUENCE field and a PIGGYBACK field, just as in data packets. The MORE and CONTROL bits are both 0. The user data field is also expanded to a maximum of 128 bytes, the first 2 of which are the *Datagram Identification*. Each Datagram is sent independently of all other datagrams. In the facilities field of a Datagram, a request can be made for explicit notification of delivery (or lack thereof). The notification packet, called a *Datagram Service Signal* packet, then tells whether the Datagram was delivered or not, and if not delivered, why not (destination address incorrect, network congestion, etc.). The Datagram is identified by the Datagram identification.

The flow control parameters do not have end-to-end meaning (i.e., D must be 0). Their purpose is to prevent the DTE from trying to send datagrams faster than the NODE can accept them. Datagrams can use any

²In other words, it is the equivalent of a Datagram service provided by the network

Virtual Circuit number not in use for a virtual call or permanent Virtual Circuit. The RESET REQUEST, RESTART REQUEST, RR, RNR, and DIAGNOSTIC packets retain their normal meaning on Datagram circuits.

In the other Datagram facility (Japanese proposal), called *fast select*, the CALL REQUEST packet is also expanded to include 128 bytes of user data, but the third byte remains the way it was. *Fast select* is requested as a facility. As far as the network is concerned, the packet really is an attempt to establish a Virtual Circuit. When fast select is used, the called DTE may reject the attempted call with a CLEAR REQUEST packet, which has also been expanded to include 128 bytes of reply data. However, it may also accept the call, in which case the Virtual Circuit is set up normally.

9.11 Internet Network Layer Protocol (IP)

The Internet Network Layer protocol is rather confusingly known as the *Internet Protocol*. Confusing, because IP is *not* the only Internet protocol, but merely the protocol used by the Internet at the Network Layer. In outline, it works as follows. The Transport Layer (TCP or UDP) takes messages and breaks them up into Datagrams of up to 64K bytes each. Each Datagram is transmitted through the Internet, possibly being fragmented into smaller units as it goes. When all the pieces finally get to the destination machine, they are re-assembled by the Transport Layer to reform the original message.

An IP Datagram consists of a *header* part and a *text* part. The header has a 20-byte fixed part and a variable length optional part. The header format is illustrated in Fig. 9.15.

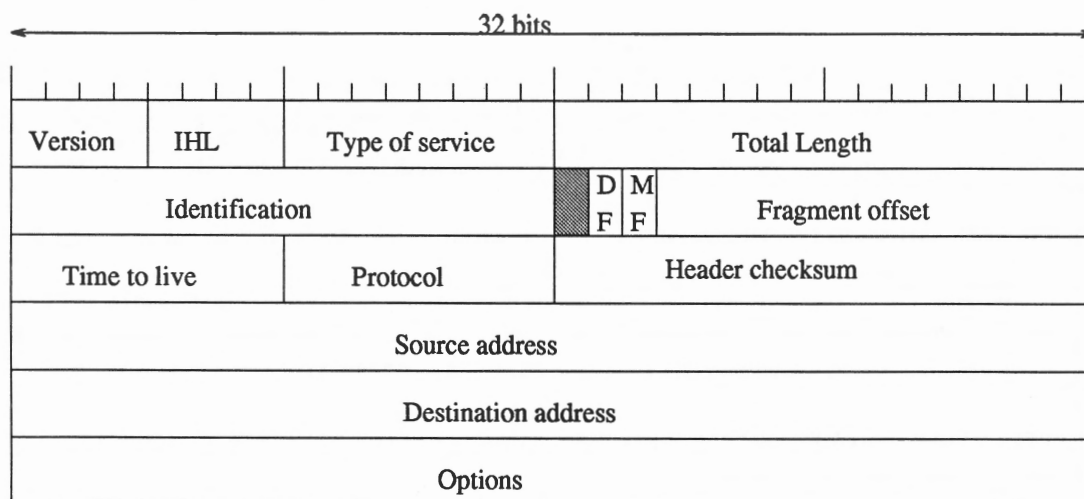


Figure 9.15: The IP (Internet Protocol) header format

- The *Version field* keeps track of which version of the protocol the Datagram belongs to. By including the version in each Datagram, the possibility of changing protocols while the network is operating is not excluded.
- Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5.
- The *Type of service* field allows the host to tell the Service Provider what kind of service it wants. Various combinations of reliability and speed are possible. For digitized voice, fast delivery is far

more important than correcting transmission errors. For file transfer, accurate transmission is far more important than speedy delivery. Various other combinations are also possible from routine traffic to flash override.

- The *Total length* includes everything in the Datagram – both header and data. The maximum length is 65,536 bytes.
- The *Identification* field is needed to allow the destination host to determine which Datagram a newly arrived fragment belongs to. All the fragments of a Datagram contain the same *identification* value.
- Next comes an unused bit and then two 1-bit fields. *DF* stands for Don't Fragment. It is an order to the gateways not to fragment the Datagram because the destination is incapable of putting the pieces back together again. For example, a Datagram being downloaded to a small personal computer for execution might be marked with *DF* because the PC's ROM expects the whole program in one Datagram. If the Datagram cannot be passed through a network, it must either be routed around the network or discarded.
- *MF* stands for More Fragments. All fragments except the last one must have this bit set. It is used as a double check against the Total length field, to make sure that no fragments are missing and that the whole Datagram is reassembled.
- The *Fragment offset* tells where in the current Datagram this fragment belongs. All fragments except the last one in a Datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per Datagram, giving a maximum Datagram length of 65,536 bytes, in agreement with the *Total length* field.
- The *Time-to-live* field is a counter used to limit packet lifetimes. When it becomes zero, the packet is destroyed. The unit of time is the second, allowing a maximum lifetime of 255 sec.
- When the network layer has assembled a complete Datagram, it needs to know what to do with it. The *Protocol* field tells which of the various transport processes the Datagram belongs to. TCP is certainly one possibility, but there may be others.
- The *Header checksum* verifies the header only. Such a checksum is useful because the header may change at a gateway (e.g., fragmentation may occur).
- The *Source address* and *Destination address* indicate the network number and host number. Four different formats are used, as illustrated in Fig. 9.16. The four schemes allow for up to 128 networks, presumably LANs, with up to 256 hosts each, and multicast, in which a Datagram is directed at a group of hosts. Addresses beginning with 1111 are reserved for future use.
- The *Options* field is used for security, source routing, error reporting, debugging, time stamping, and other information. Basically it provides an escape to allow subsequent versions of the protocol to include information not present in the original design, to allow experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed.

The operation of the Internet is monitored closely by the NODEs and gateways. When something suspicious occurs, the event is reported by the *ICMP* (*Internet Control Message Protocol*), which is also used to test the Internet. About a dozen types of ICMP messages are defined. Each message type is encapsulated in an IP packet.

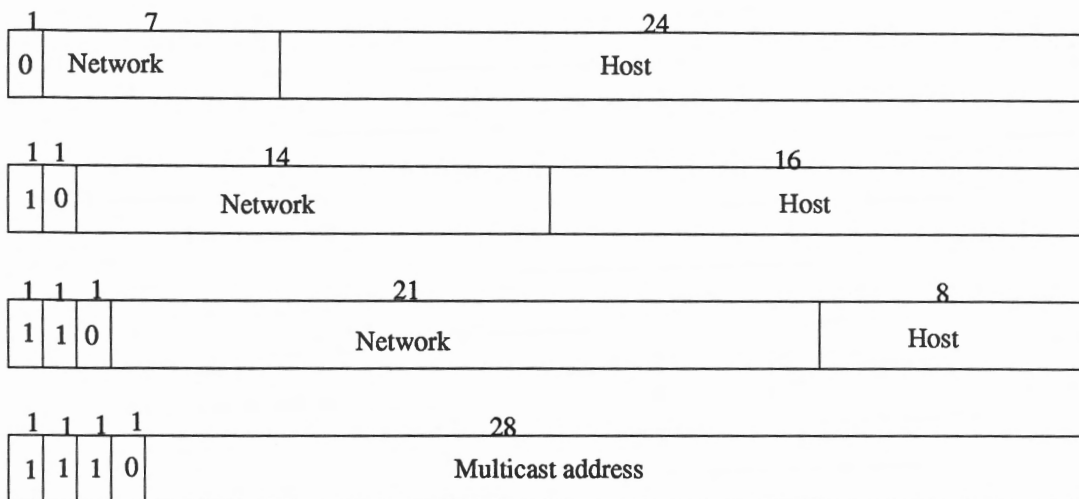


Figure 9.16: Source and destination address formats in the IP header

The *DESTINATION UNREACHABLE* message is used when the Service Provider or a gateway cannot locate the destination, or a packet with the DF bit cannot be delivered because a "small- packet" network stand in the way.

The *TIME EXCEEDED* message is sent when a packet is dropped due to its counter reaching zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.

The *PARAMETER PROBLEM* message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software, or possibly in the software of a gateway transited.

The *SOURCE QUENCH* message is used to throttle hosts that are sending too many packets. When a host receives this message, it is expected to slow down.

The *REDIRECT* message is used when a gateway notices that a packet seems to be routed wrong. For example, if a gateway in Los Angeles sees a packet that came from New York and is headed for Boston, this message would be used to report the event and help get the routing straightened out.

The *ECHO REQUEST* and *ECHO REPLY* message are used to see if a given destination is reachable and alive. Upon receiving the *ECHO* message, the destination is expected to send an *ECHO REPLY* message back. The *TIMESTAMP REQUEST* and *TIMESTAMP REPLY* message are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

In addition to these messages, there are four others that deal with internet addressing, to allow hosts to discover their network numbers and to handle the case of multiple LANs sharing a single IP address.

9.12 Asynchronous Transfer Mode (ATM) Networks

Asynchronous Transfer Mode (ATM), also known as *cell relay* or *cell switching* takes advantages of the low error rates of modern digital circuits to provide faster packet switching than X.25. ATM is a connection-oriented protocol, in other words, between every sender and receiver, there is a set route that the cells follow, and these cells are guaranteed to arrive in the same sequence as they were sent.

The *Asynchronous Transfer Mode (ATM)* allows various sources of traffic, which are likely to produce bit streams at different rates, to be packetised into cells which are then multiplexed onto the single ATM digital pipe of capacity between 125.6Mbps and 622.08Mbps as illustrated in Fig. 9.17.

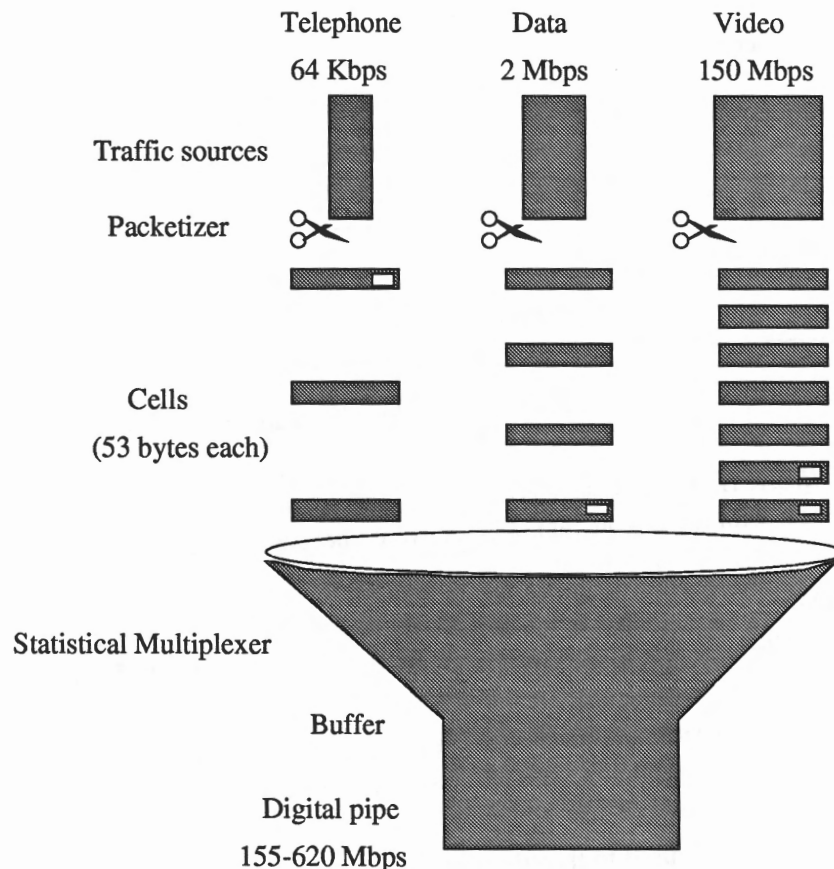


Figure 9.17: The ATM principle

Asynchronous transfer mode (ATM) is in some ways similar to packet switching using X.25 and to frame relay. Like packet switching and frame relay, ATM involves the transfer of data in discrete chunks. Also, like packet switching and frame relay, ATM allows multiple logical connections to be multiplexed over a single physical interface. In the case of ATM, the information flow on each logical connection is organized into fixed-size packets, called cells.

ATM is a streamlined protocol with minimal error and flow control capabilities. This reduces the overhead of processing ATM cells and reduces the number of overhead bits required with each cell, thus enabling ATM to operate at high data rates. Further, the use of fixed-size cells simplifies the processing required at each ATM node, again supporting the use of ATM at high data rates.

Two layers of the protocol architecture relate to ATM functions. There is an *ATM layer* common to all services that provides cell transfer capabilities, and *ATM adaptation layer (AAL)* that is *service dependent*. The ATM layer defines the transmission of data in fixed size cells and also defines the use of logical connections. The use of ATM creates the need for an adaptation layer to support information transfer protocols not based on ATM. The AAL maps higher-layer information into ATM cells to be transported over an ATM network, then collects information from ATM cells for delivery to higher layers.

In ATM networks logical links are referred to as *Virtual Channel Connections (VCCs)*. A VCC is analogous

to a Virtual Circuit in X.25; it is the basic unit of switching in an ATM network. A VCC is set up between two end users through the network and a variable-rate, full-duplex flow of fixed-size cells is exchanged over the connection. VCCs are also used for user-network exchange (control signaling) and network-network exchange (network management and routing).

For ATM, a second sublayer of processing has been introduced that deals with the concept of Virtual Path as illustrated in Fig. 9.18. A *Virtual Path Connection (VPC)* is a bundle of VCCs that have the same endpoints. Thus, all of the cells flowing over all of the VCCs in a single VPC are switched together.

The virtual path concept was developed in response to a trend in high-speed networking in which the control cost of the network is becoming an increasingly higher proportion of the overall network cost. The virtual path technique helps contain the control cost by grouping connections sharing common paths through the network into a single unit. Network management actions can then be applied to a small number of groups of connections instead of a large number of individual connections.

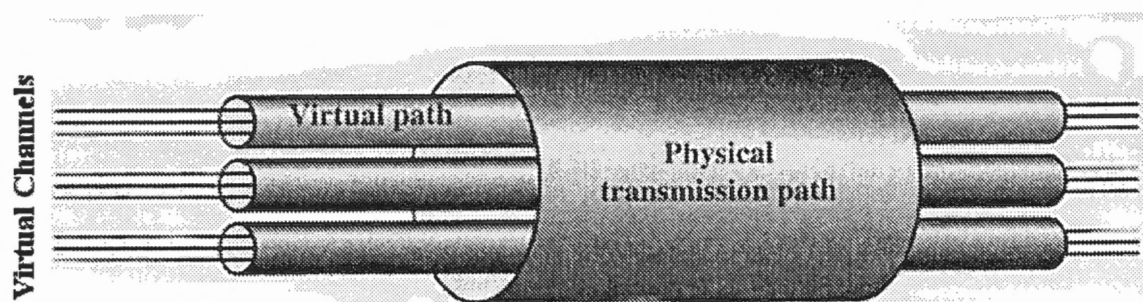


Figure 9.18: ATM connection paths

9.12.1 Quality of Service (QoS)

By now it should be clear that ATM provides a connection oriented service or a virtual circuit service as we called it in Sec. 9.4 on page 140. Fig. 9.19 suggests a general algorithm for the call establishment process using Virtual Channels and Virtual Paths. An important concepts in the call establishment process is the Quality of Service specified by parameters such as the cell loss ratio, the cell transfer delay and the cell block error ratio.

1. *The cell loss ratio.* The ratio of cells that are lost for each connection, based on an end-to-end measurement. The Cell Loss Priority has a great effect on whether a cell is discarded in cases of high congestion.
2. *The cell transfer delay.* The time it takes a single cell to traverse the network between source and destination. The tolerance of cells to delay and variations in delay is crucial to many applications. In the switch nodes, traffic of different service classes is separated so that cell transfer delay is appropriate to the particular class of traffic. Constant bit-rate traffic receives the highest priority in this case.
3. *The cell block error ratio.* The total number of blocks that contain an error divided by the total number of cells transmitted in that period. Note that this is distinct from the single error ratio, since single errors are picked up by the ATM adaption layer using the Header Error Control (HEC) field, and corrected without retransmission. By contrast, block errors force the retransmission of the entire cell.

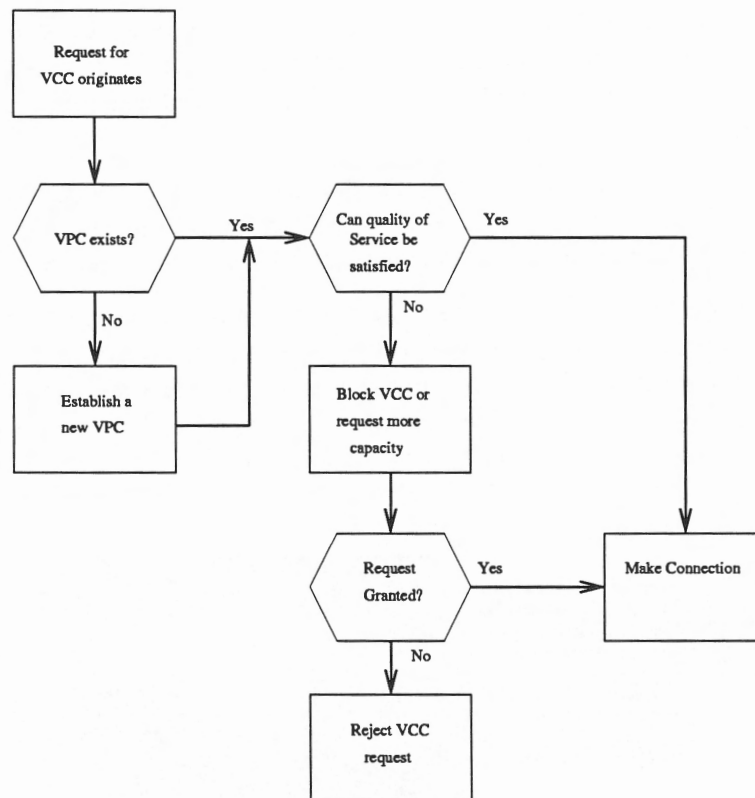


Figure 9.19: ATM call establishment

The toleration of the various services to errors is crucial here: data needs a low cell block error ratio whereas voice can be transmitted with a higher cell block error ratio.

The types of traffic parameters that can be negotiated include average rate, peak rate, burstiness, and peak duration. The network may need a number of strategies to deal with congestion and to manage existing and requested VCCs. At the crudest level, the network may simply deny new requests for VCCs to prevent congestion. Additionally, cells may be discarded if negotiated parameters are violated or if congestion becomes severe. In an extreme situation, existing connections might be terminated.

9.12.2 ATM Cells

The asynchronous transfer mode makes use of fixed-size cells, consisting of a 5-octet header and a 48-octet information field. The peculiar choice was again a political compromise between the Americans and the Europeans who wanted different cell sizes. There are several advantages to the use of small, fixed-size cells. First, the use of small cells may reduce queueing delay for a high-priority cell, because it waits less if it arrives slightly behind a lower-priority cell that has gained access to a resource (e.g., the transmitter). Second, it appears that fixed-size cells can be switched more efficiently, which is important for the very high data rates of ATM. With fixed-size cells, it is easier to implement the switching mechanism in hardware.

Fig. 9.20(a) shows the cell header format at the user-network interface and Fig. 9.20(b) shows the cell header format internal to the network.

The *Generic Flow Control (GFC)* field does not appear in the cell header internal to the network, but only at the user-network interface. Hence, it can be used for control of cell flow only at the local user-network

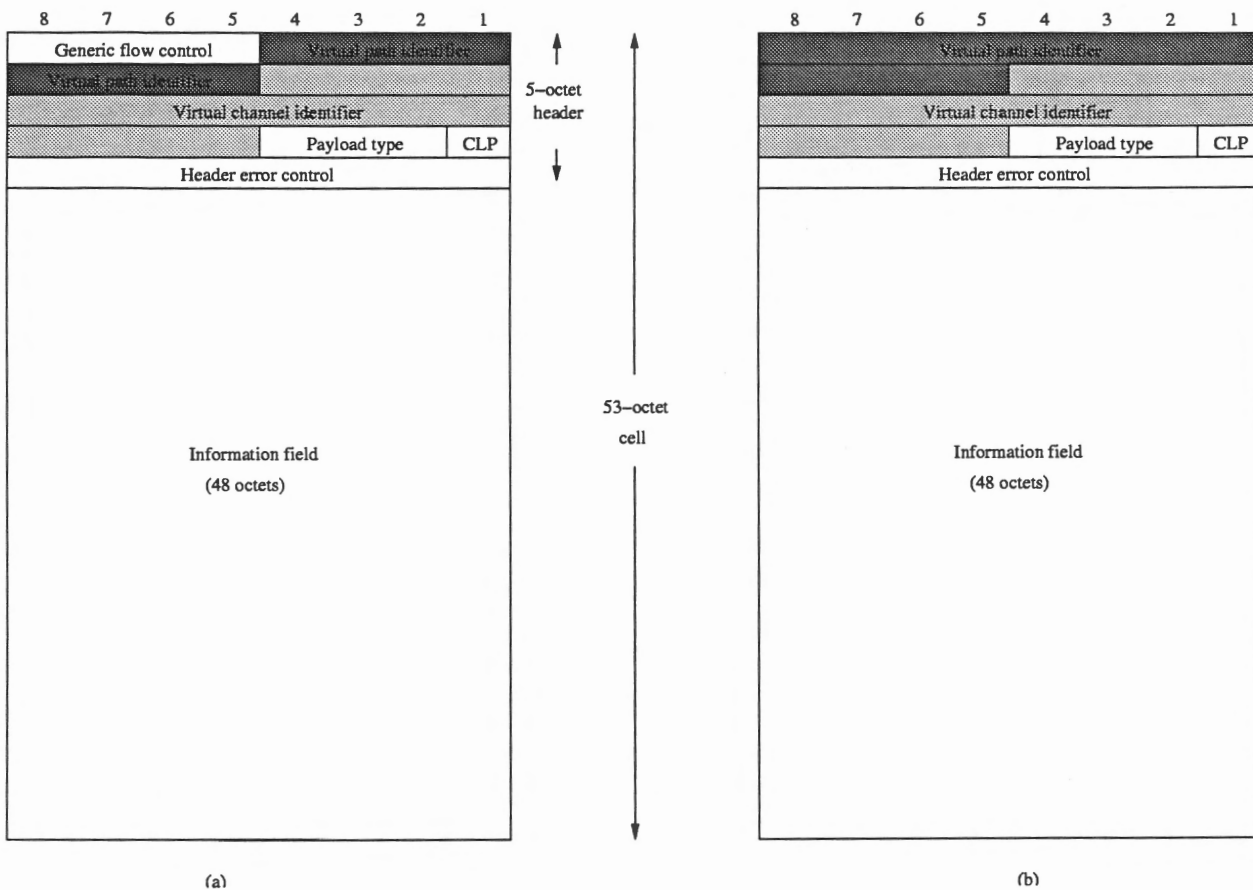


Figure 9.20: ATM cell format at (a) User-Service Provider interface and (b) internal to the network

interface. The field could be used to assist the user in controlling the flow of traffic for different QoS. In any case, the GFC mechanism is used to alleviate short-term overload conditions in the network.

The *Virtual Path Identifier (VPI)* constitutes a routing field for the network. It is 8 bits long at the user-network interface and 12 bits long at the network-network interface. The latter allows support for an expanded number of VPCs internal to the network, to include those supporting subscribers and those required for network management. The virtual channel identifier (VCI) is used for routine to and from the end user.

The *Payload Type (PT)* field indicates the type of information in the information field. Table 9.4 shows the interpretation of the PT bits.

The *Cell Loss Priority (CLP)* bit. Some services, like data transmission, value accurate transmission more highly than the connection-oriented services. For this reason, the CLP bit is included, to enable a priority to be assigned to the integrity of data in the cell being transmitted. When the network becomes congested, those cells with a low priority (CLP = 1) may be discarded to protect the high-priority (CLP = 0) traffic.

The 8-bit *Header Error Control (HEC)* field in the header performs simple error correction for the frame (error correction is discussed in Sec. 6.4 on page 80). Implementation of the HEC significantly improves the performance of the ATM networks.

Pay load type	Meaning
000	User data cell, no congestion, cell type 0
001	User data cell, no congestion, cell type 1
010	User data cell, experienced, cell type 0
011	User data cell, experienced, cell type 0
100	Maintenance information between adjacent switches
101	Maintenance information between source and destination switches
110	Resource Management cell (used for ABR congestion control)
111	Reserved for future use

Table 9.4: Values of the Pay Load type (PT) field in an ATM header cell

9.12.3 ATM Service Classes

An ATM network is designed to be able to transfer many different types of traffic simultaneously, including real-time flows such as voice, video, and bursty packet flows. Although each such traffic flow is handled as a stream of 53-octet cells traveling through a virtual channel, the way in which each data flow is handled within the network depends on the characteristics of the traffic flow and the requirements of the application. For example, real-time video traffic must be delivered within minimum variation in delay.

In this section, we summarize ATM service categories, which are used by an end system to identify the type of service required. The following service categories have been defined by the ATM standards committee known as the ATM Forum:

1. Real-time service.
 - Constant bit rate (CBR).
 - Real-time variable bit rate (rt-VBR)
2. Non-real-time service.
 - Non-real-time variable bit rate (nrt-VBR).
 - Available bit rate (ABR).
 - Unspecified bit rate (UBR).

9.12.4 Real-Time Services

The most important distinction among applications concerns the amount of delay and the variability of delay, referred to as jitter, that the application can tolerate. Real-time application typically involve a flow of information to a user that is intended to reproduce that flow at a source. For example, a user expects a flow of audio or video information to be presented in a continuous, smooth fashion. A lack of continuity or excessive loss results in significant loss of quality. Applications that involve interaction between people have tight constraints on delay. Typically, any delay above a few hundred milliseconds becomes noticeable and annoying. Accordingly, the demands in the ATM network for switching and delivery of real-time data are high.

- *Constant Bit Rate (CBR)* The CBR service is perhaps the simplest service to define. It is used by applications that require a fixed data rate that is continuously available during the connection lifetime

and a relatively tight upper bound on transfer delay. CBR is commonly used for uncompressed audio and video information. Example of CBR applications include the following:

- Videoconferencing.
 - Interactive audio (e.g., telephony).
 - Audio/video distribution (e.g., television, distance learning, pay-per-view).
 - Audio/video retrieval (e.g., video-on-demand, audio library)
- *Real-Time Variable Bit Rate (rt-VBR)* The rt-VBR category is intended for time-sensitive applications; that is, those requiring tightly constrained delay and delay variations. The principal difference between applications appropriate for rt-VBR and those appropriate for CBR is that rt-VBR applications transmit at a rate that varies with time. Equivalently, an rt-VBR source can be characterized as somewhat bursty. For example, the standard approach to video compression results in a sequence of image frames of varying sizes. Because real-time video requires a uniform frame transmission rate, the actual data rate varies.

The rt-VBR service allows the network more flexibility than CBR. The network is able to statistically multiplex a number of connections over the same dedicated capacity and still provide the required service to each connection.

- *Non-Real-Time Services.* Non-real-time services are intended for applications that have bursty traffic characteristics and do not have tight constraints on delay and delay variation. Accordingly, the network has greater flexibility in handling such traffic flows and can make greater use of statistical multiplexing to increase network efficiency.
- *Non-Real-Time Variable Bit Rate (nrt-VBR).* For some non-real-time applications, it is possible to characterize the expected traffic flow so that the network can provide substantially improved quality of service (QoS) in the areas of loss and delay. Such applications can use the nrt-VBR service. With this service, the end system specifies a peak cell rate, a sustainable or average cell rate, and a measure of how bursty or clumped the cells may be. With this information, the network can allocate resources to provide relatively low delay and minimal cell loss.

The nrt-VBR service can be used for data transfers that have critical response-time requirements. Examples include airline reservations, banking transactions, and process monitoring.

- *Unspecified Bit Rate (UBR).* At any given time, a certain amount of the capacity of an ATM network is consumed in carrying CBR and the two types of VBR traffic. Additional capacity is available for one or both of the following reasons:
 1. Not all of the total resources have been committed to CBR and VBR traffic, and
 2. The bursty nature of VBR traffic means that at some times less than the committed capacity is being used.

All of this unused capacity could be made available for the UBR service. This service is suitable for applications that can tolerate variable delays and some cell losses, which is typically true of TCP-based traffic. With UBR, cells are forwarded on a first-in-first-out (FIFO) basis using the capacity not consumed by other services; both delays and variable losses are possible. No initial commitment is made to a UBR source and no feedback concerning congestion is provided; this is referred to as a best-effort service. Examples of UBR applications include the following:

- Text/data/image transfer, messaging, distribution, retrieval.

– emote terminal (e.g., telecommuting)

- *Available Bit Rate (ABR)*. Bursty applications that use a reliable end-to-end protocol such as X.25 can detect congestion in a network by means of increased round-trip delays and packet discarding. However, X.25 has no mechanism for causing the resources within the network to be shared fairly among many Virtual Circuits. Further, it does not minimize congestion as efficiently as is possible using explicit information from congested nodes within the network.

To improve the service provided to bursty sources that would otherwise use UBR, the ABR service has been defined. An application using ABR specifies a peak cell rate (PCR) that it will use and a minimum cell rate (MCR) that it requires. The network allocates resources so that all ABR applications receive at least their MCR capacity. Any unused capacity is then share in a fair and controlled fashion among all ABR sources. The ABR mechanism uses explicit feedback to sources to assure that capacity is fairly allocated. Any capacity not used by ABR sources remains available for UBR traffic.

Finally, as far as ATM is concerned, it is necessary to know that ATM requires an *adaption layer* to function with protocols not based upon ATM. Two such examples are PCM (Pulse Code modulation, see Sec. 2.5.3 on page 39) and the Internet Protocol. PCM voice is an application that produces a stream of bits from digitizing a voice signal. To employ this application over ATM, it is necessary to assemble PCM bits into cells for transmission and to re-assemble them into radio voice signals on reception.

In the case of IP over ATM, IP packets have to be mapped onto ATM cells which, because of the IP packets sizes, implies segmenting IP packets into more than one cell and re-assembling the cells on reception. Using IP over ATM, all the existing Internet infrastructure can be used over an ATM network.

9.13 Exercises

Exercise 9.1 *Explain where the ATM data rate of 155 Mbps comes from.*

Exercise 9.2 *What factors effect the quality of a ATM network service?*

Exercise 9.3 *How does ATM cater for multimedia?*

Exercise 9.4 *What services are provided by the Network Layer?*

Exercise 9.5 *Contrast a Virtual Circuit and a Datagram service. Include in the comparision the pros and cons.*

Exercise 9.6 *Show the NODE tables at nodes A, C, G and H in Fig. 9.21 after the following operations:*

- *A establishes a Virtual Circuit with C.*
- *B establishes a Virtual Circuit with H.*
- *F establishes a Virtual Circuit with C*

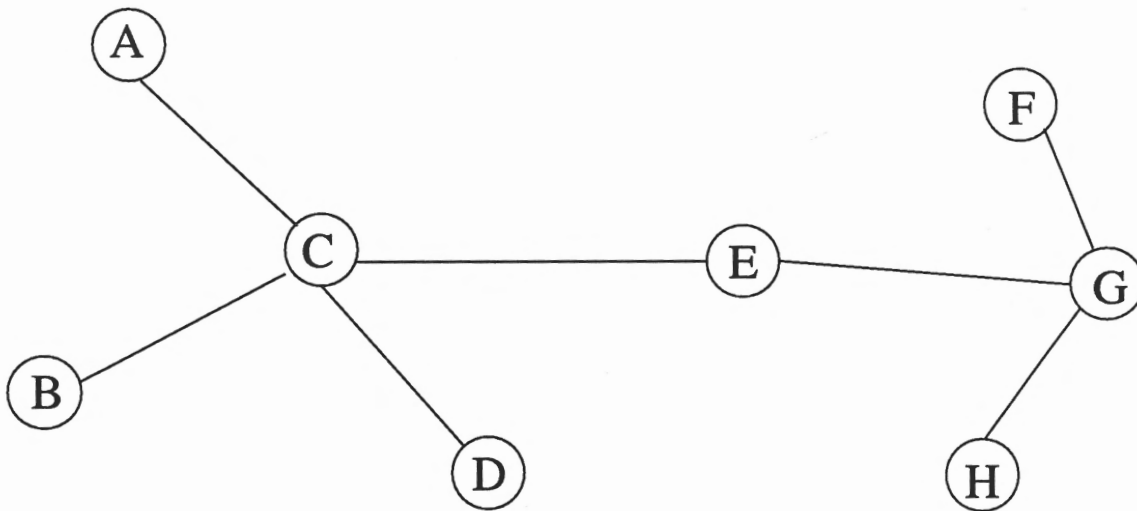


Figure 9.21: Virtual circuit configuration.

- *The Virtual Circuit from A to C ends successfully.*
- *D establishes a Virtual Circuit with A.*
- *C establishes a Virtual Circuit with A.*
- *H establishes a Virtual Circuit with F.*
- *The Virtual Circuit F established with C times out.*
- *F establishes a Virtual Circuit with B.*
- *E establishes a Virtual Circuit with B.*
- *A establishes a Virtual Circuit with G.*

What would happen if E were to go down? How could this be avoided?

Exercise 9.7 *How can routing algorithms be classified?*

Exercise 9.8 *When referring to routing algorithms, we talk of decision place, decision time and information source. Explain how these characteristics differ between the two classes of routing algorithms.*

Exercise 9.9 *Explain the difference between congestion and deadlock.*

Exercise 9.10 *If you have flow control why do you need deadlock prevention?*

Exercise 9.11 *What is deadlock and how can it be prevented?*

Exercise 9.12 *What are the three layers in the X.25 protocol? What is the purpose of each layer? How do these layers relate to the OSI model?*

Exercise 9.13 *True or False : If a network is using the X.25 protocol, then it's Service Provider uses virtual circuits internally. Motivate your answer.*

Exercise 9.14 *What is the purpose of the Time-to-live field in an IP packet?*

Chapter 10

Transport Layer

10.1 Objectives of this Chapter

The transport layer contains the first protocols which are truly user-oriented in that, while using the underlying network layer, its function is to provide the user with a reliable, error-free end-to-end service.

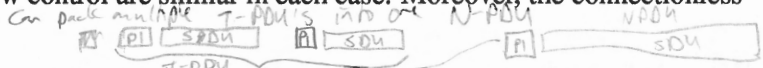
There are many such protocols. The ones we shall cover in outline only in this chapter, are:

1. ISO OSI protocol layer including connection set up and flow control.
2. TCP the transport layer of the Internet.
3. Sockets, as they are provided for by the Unix operating system, and
4. the ATM Adaption Layer (or AAL) protocols.

10.2 Introduction

The transport protocol is the interface between the *network service provider* and the user of the network services. For that reason it is an end-to-end protocol which, as those words imply, ensures that the end-users receive the services it provides without the user having to concern himself with the quality of the underlying network. In other words, whether the underlying network uses a connection oriented X.25 protocol or a connectionless IP protocol concerns the transport layer only to the extent that it will have to adjust what it does depending on what service it itself offers the user.

Just as there are two types of network service, the transport layer protocols also provide for a connection-oriented and a connectionless service to the user. The connection-oriented transport service is similar to the connection-oriented network service in that in both cases connections have three phases: establishment, data transfer and disconnect. Addressing and flow control are similar in each case. Moreover, the connectionless service is very similar as well.



If the two services are so similar, why bother? Think of the answer before you proceed. The user normally requires a certain *quality of service (QoS)* independent of what the network offers. For instance, what happens if the network layer offers connection-oriented service but it is unreliable in that it loses packets or crashes from time to time. By ensuring its own QoS the transport layer can recover halfway through a long transmission if the network connection is suddenly lost. It would do so by setting up a *new* network connection, ask its remote peer what data was received correctly and re-synchronize.

In general the following are some of the functions required from the transport layer:

1. *Connection management.* This function defines the rules that allow two users to start communicating. Certain parameters such as window size, message lengths and so on are agreed in a phase also called *handshaking*.
2. *Flow control.* Flow control limits the number of messages that one user can send to the other without waiting for an acknowledgement. Note that this is the third layer at which we meet flow control which again seems redundant and which would be in a perfect world. Apart from the reasons already mentioned, just as the network layer may have more than one connection over a single link layer, the independent transport layer must have its own flow control mechanism for that reason as well (it is called "*multiplexing*" in network terminology).
3. *Error detection.* Remember, if the underlying network service is a datagram (or connectionless) service, it provides no guarantee to the next (transport) layer about the sequence in which packets are delivered or whether there may be duplicates or not; this is up to the transport layer.

In the following section we discuss each of the first two functions as they are defined in the ISO Transport Protocol (or ISO TP). First, however, we discuss some of the general characteristics of the ISO Transport Protocol.

10.3 ISO Transport Protocol

The transport layer provides transport services through the transport services access point (TSAP) to the TS User (refer to Fig. 1.8, page 15) while itself using the services of the underlying network layer through an associated network service access point (NSAP). Collectively, it is the TSAP and NSAP addresses that help to identify, uniquely, the application using the transport connection.

This emphasis on the services *used* and the services *provided* are central to the concept of the ISO OSI protocols first described in Sec.1.7 on page 12 of these notes. Indeed, the specification of each protocol layer comprises of two sets of documents: a *service definition document* and a *protocol specification document*. The service definition document contains a description of the services provided by the layer to the one above it — that is the *user services*. Normally, these take the form of a defined set of *service primitives*, each with an associated set of *service parameters* (much like a function call) . It is through the service parameters that the layer above initiates the transfer of information to the *peer layer* at the remote site.

The protocol specification document in turn contains:

1. A precise definition of the *Protocol Data Units (PDUs)* used by the protocol entity of that layer to communicate with the peer protocol entity at the remote system.
2. A specification of the services used by the layer below it (the network layer in the case of the transport protocol) to transfer each PDU type.
3. A precise definition of the protocol entity presented in one of the formal definition techniques (FDTs) such as the Specification and Description Language (SDL) (discussed in Volume II of these notes). While this is ideally the case, this writer has yet to see an FDT description of any of the ISO protocols.

As already mentioned, the function of the transport layer is to provide the application, or theoretically, the session layer with a reliable, error-free, in-sequence, with no loss or duplication, message transport facility that is independent of the quality of the underlying network layer. ISO identifies five classes of network service with which the transport protocol must cope:

1. *Class 0*. Simple class assuming a highly reliable PSDN network.
2. *Class 1*. Basic error recovery class.
3. *Class 2*. Assuming multiple transport connections over a single network connection, also called “multiplexing”
4. *Class 3*. Error recovery and multiplexing class.
5. *Class 4*. Error detection and recovery class. This class has the maximum functionality, catering for error detection, flow control, etc. It is meant to work with unreliable connectionless network layers.

Depending on the quality of service (QoS) of the network class, the transport layer protocols come in various flavours. The one which can handle Class 4 networks, in other words it assumes the worst, is known as Transport Layer Class 4 or simply ISO TP4. It is a long and complicated document with forward and backward cross references describing, *inter alia* 3 different timers, all of which provides a real challenge to the implementor. We shall obviously not go into such detail, but the connection establishment and flow control mechanisms in TP4 are general enough to discuss in more detail.

10.3.1 Establishing a connection

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a *CONNECTION REQUEST (CR) TPDU* to the destination and wait for a *CONNECTION ACCEPTED (CA)* reply. The problem occurs when the network can and does lose, store and duplicate packets.

Imagine a subnet that is so congested that acknowledgements hardly ever get back in time, and each packet times out and is retransmitted two to three times. Suppose that the subnet uses datagrams inside, and every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the subnet and take a long time to arrive, that is, they are stored in the subnet and pop out much later.

The worst possible nightmare is as follows. A user established a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person, and then releases the connection. Unfortunately, each packet in the scenario is duplicated and stored in the subnet. After the connection has been released, all the packets pop out of the subnet and arrive at the destination in order, asking the bank to establish a new connection, transfer money (again), and release the connection. The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

There is no satisfactory solution to the delayed duplicate problem and the methods which are in use all ensure that no packet lives longer than some known time. Packet lifetime can be restricted in one of the following ways:

1. Restricted network design.
2. Packet hop counters.
3. Timestamping each packet

The first technique includes methods which prevent packets from looping combined with some way of bounding congestion delay over the longest path. Clearly this approach only works if the topology of the network is known and stable; none which is true in the case of the Internet.

The second method obviously works by every node incrementing the hop count in a packet every time the packet is forwarded. A packet is simply discarded when its hop count exceeds a known maximum. Again, this method assumes that the topology of the network is known to the point that a maximum hop count can be determined in every case.

The last technique requires each packet to carry the time it was created or, equivalently, the time it has been alive. A router simply kills a packet whose time-to-live has been exceeded. If we think about it carefully however, it implies all the router clocks in the network are synchronized, and stay synchronized even if the router goes down. This is a non-trivial task, particularly when the network spans more than one continent and time zone. The latter is made easier nowadays that most countries have a radio signal transmitting an accurate atomic clock time to which every device (and person) can be synchronized. With a combination of this global clock time and sequence numbering, the delayed duplicate problem can be solved for data TPDU's, but a connection, which is vulnerable to the same problem, first has to be established.

Tomlinson[Tom75] introduced the three-way handshake to solve the problem of delayed duplicates during connection setup. The normal setup procedure when Host 1 initiates is shown in Fig. 10.1(a). Host 1 chooses a sequence number, x , and sends a *CONNECTION REQUEST TPDU* containing it to Host 2. Host 2 replies with a *CONNECTION ACCEPTED TPDU* acknowledging x and announcing its own initial sequence number, y . Finally, Host 1 acknowledges choice of an initial sequence number from Host 2 in the first data TPDU that it sends.

Now let us see how the three-way handshake works in the presence of delayed duplicate control TPDU's. In Fig. 10.1(b), the first TPDU is a delayed duplicate *CONNECTION REQUEST* from an old connection. This TPDU arrives at Host 2 without Host 1 knowing about it. Host 2 reacts to this TPDU by sending Host 1 a *CONNECTION ACCEPTED TPDU*, in effect asking for verification that Host 1 was indeed trying to set up

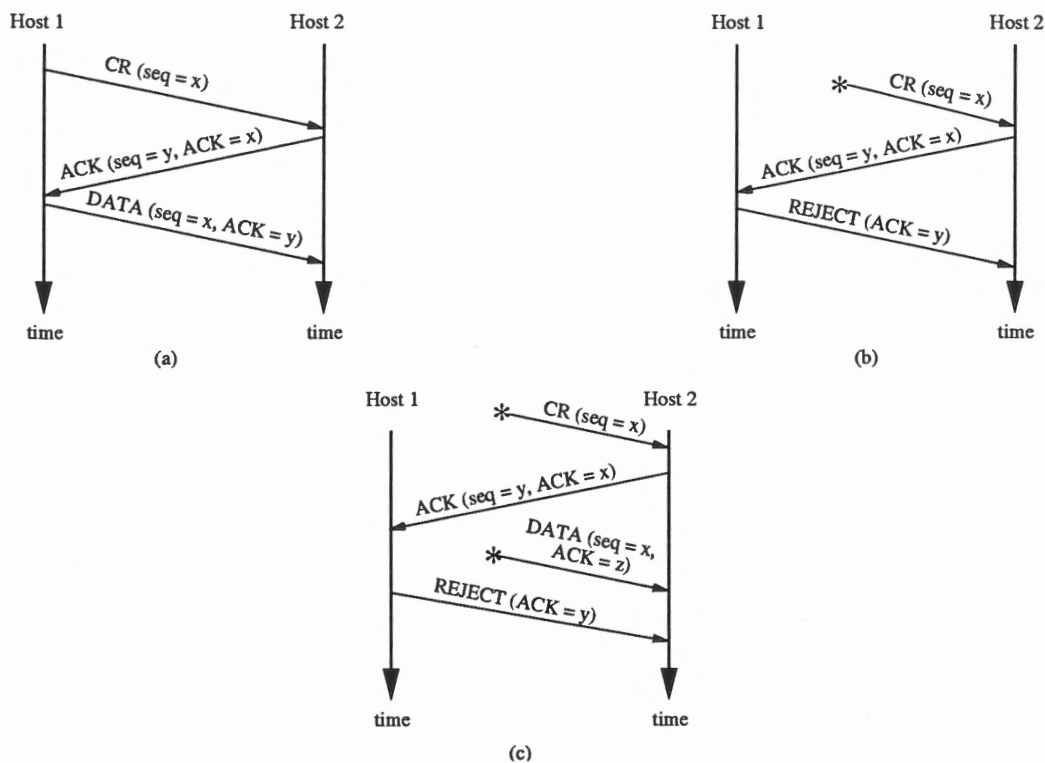


Figure 10.1: Establishing a connection using a three-way handshake

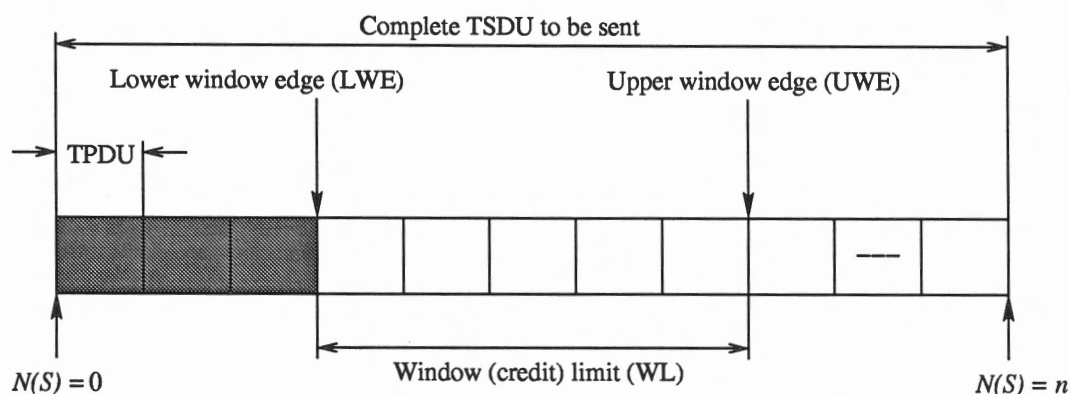
a new connection. When Host 1 rejects Host 2's attempt to establish the connection, Host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

The worst case is when both a delayed *CONNECTION REQUEST* and an acknowledgement to a *CONNECTION ACCEPTED* are floating around in the subnet. This case is shown in Fig. 10.1(c). As in the previous example, Host 2 gets a delayed *CONNECTION REQUEST* and replies to it. At this point it is crucial to realize that Host 2 has proposed using y as the initial sequence number for Host 2 to Host 1 traffic, knowing full well that no TPDUs containing sequence number y or acknowledgements to y are still in existence. When the second delayed TPDU arrives at Host 2, the fact that z has been acknowledged rather than y tells Host 2 that this, too, is an old duplicate. The important thing to realize here is that there is no combination of old *CONNECTION REQUEST*, *CONNECTION ACCEPTED*, or other TPDUs that can cause the protocol to fail and have a connection set up by accident when no one wants it.

10.3.2 Flow control in the transport layer

The objective of a flow control mechanism is to limit the amount of data (or DT-TPDUs) transmitted by the sending transport entity to a level that the receiver can accommodate. Clearly, therefore, if the transport entity is only servicing a single user, the appropriate amount of buffer storage required to process the subsequent user TPDUs may be reserved in advance. Consequently, when the transport connection is being established no flow control mechanism needs to be provided. If the transport entity is servicing multiple users, however, and buffer storage is reserved on a statistical basis, then a flow control mechanism must be supported by the protocol. This again is determined by the class of service provided by the transport entity.

The flow control mechanism used with TP 4 is based on a variation of the sliding window protocol. An



Note:

LWE is initialised to zero and is incremented as each AK-TPDU is received.

UWE is initialised to the CDT value agreed when the connection is established and is subsequently incremented by the CDT value in each AK-TPDU received.

Flow is stopped if $N(S)$ reaches UWE.

Figure 10.2: Flow control mechanism

initial credit value, equal to the number of outstanding (unacknowledged) DT-TPDUs, for each direction of transmission is specified in the *CDT* field (refer to Fig. 10.2) of each CR-TPDU and the CC-TPDU exchanged during connection establishment. The initial sequence number for each direction of transmission is set to zero when the connection is first established and this becomes the lower window edge (*LWE*). The sender can continuously compute the upper window edge (*UWE*) by adding to the *LWE*, the credit value for the connection, modulo the size of the received sequence field. The flow of DT-TPDUs is then stopped if the send sequence number, $N(S)$, becomes equal to the *UWE* value. The *LWE* is continuously incremented as AK-TPDUs for outstanding DT-TPDUs are received. This is shown in Fig. 10.2.

The actual number of new DT-TPDUs which can be transmitted by the sender is completely under the control of the receiver and may vary during the lifetime of a connection. In addition to a receive sequence number, each AK-DTPU contains a new credit value which specifies the number of new TPDUs that the receiver is prepared to accept after the one being acknowledged. If this is zero, the sender must cease transmission of DT-TPDUs over the connection.

Normally, however, the credit value is used in the situation where the receiver allocates a fixed number of buffers for the connection. Then, as each TPDU is received, this progressively reduces the number of new TPDUs it is prepared to accept (the *UWE*) as the transfer proceeds and buffers start to be used up.

10.4 Internet Transport protocols

We mentioned in Chapter 1 that while the bureaucrats were haggling over the ISO OSI protocol standards, the world moved forward and developed the Internet protocols. The Internet transport layer protocol is widely used and known as the *Transmission Control Protocol*, or TCP for short. TCP is different from TP4 in that it serves the application protocols (such as FTP, SNMP) directly without the intermediate session and presentation layers that the OSI standards nominally prescribe.

Not surprisingly however, both protocols perform similar functions. There is a best-effort, connectionless protocol called *User Datagram Protocol (UDP)* as well as a reliable, connection oriented protocol or *Transmission Control Protocol (TCP)*. Somewhat confusingly, the Internet suite of protocols are known as the *TCP/IP* protocols, making no mention of the name of UDP or the several Internet application protocols.

The position of UDP and TCP in relation to the Internet network protocol (IP) and the protocols which use them, is illustrated in Fig. 10.3. Note that the Internet transport layer always uses the IP protocol discussed in Volume I.

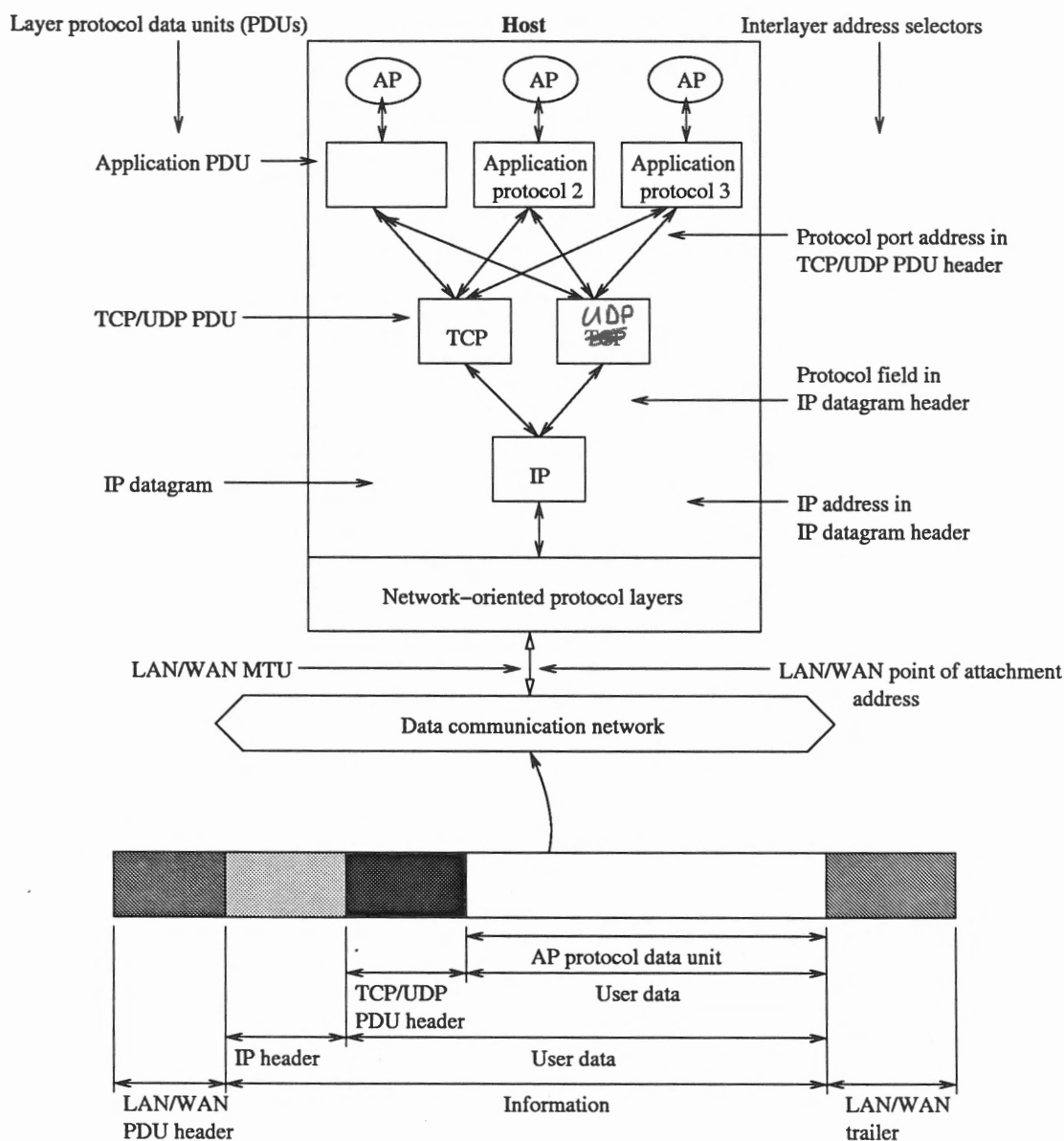


Figure 10.3: Internet transport protocols TCP and UDP in context

10.4.1 User Datagram Protocol (UDP)

UDP is a connectionless protocol with a single TPDU transported in the user data field of an IP datagram. In order to distinguish between an IP datagram and the datagram in the user field associated with UDP, the

term *user datagram* is used to refer to the latter.

With TCP/IP, the composite address of an application protocol is the Internet-wide IP address of the host plus the *port address* (or TSAP in ISO terminology). In dot notation, an example address of an application protocol is typically

128.3.2.3,53

where the first part being the IP address (network ID being 128.3 and host ID being 2.3) and 53 being the port address. In ISO terminology, the composite address is known as the *fully-qualified address*. The *source port* which can be found in the UDP header is obviously the (optional) address of the sender application and the *destination port* that of the receiver. The source port is only given if a response is required. Both are 16-bit integers. Note that locating the IP address and port address of a destination protocol is not the responsibility of UDP, or TCP for that matter.

10.4.2 Transmission Control Protocol (TCP)

TCP builds on IP using CRCs, sequence numbers, time-outs and retransmissions to create a reliable, stream oriented network. In other words, data that are error-free, with no packet losses or duplicates and in the same order as they were submitted. The work "*stream*" is used with TCP since it treats all the user data associated with a connection as duplex data streams comprising a string/stream of octets.

TCP achieves its reliable service by transmitting all data in units known as *segments* whose layout is illustrated in Fig. 10.4. Normally the TCP protocol decides when a new segment is transmitted. At the destination, the receiver buffers the segments in a buffer associated with the application and delivers it when the buffer is full. A segment can thus consist of multiple user messages if the messages exchanged are short or, in the case of a large file transfer, only part of the full message unit. The maximum length of each segment is a function of the TCP protocol and can be as large as 2_{16} or 65536 bytes. Other facilities which exist

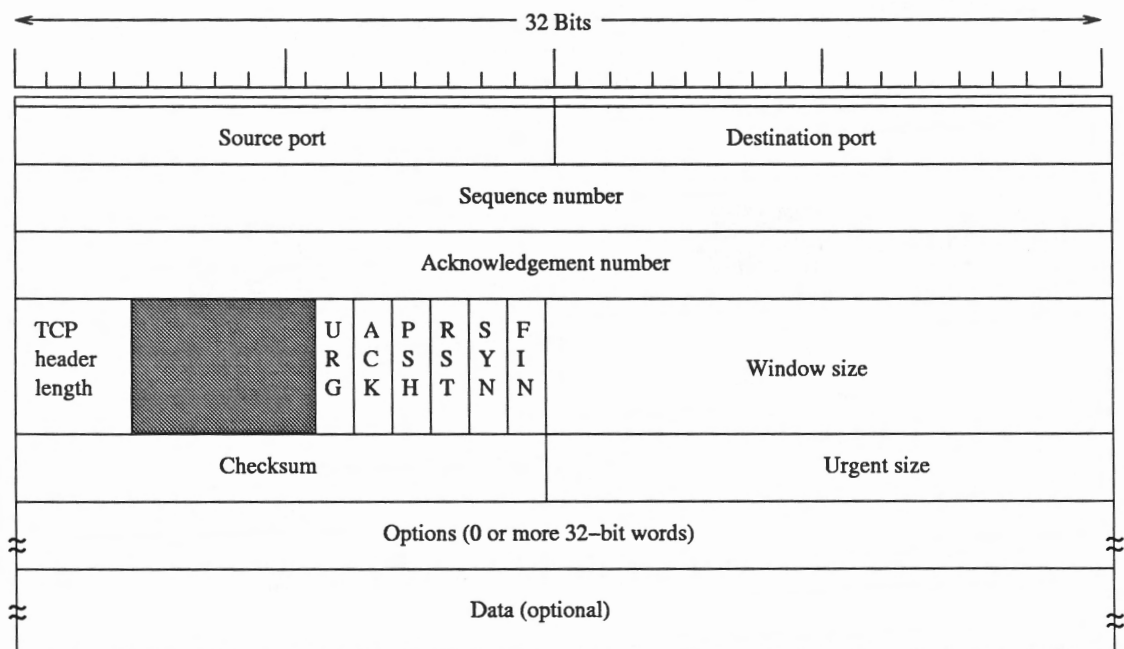


Figure 10.4: TCP segment layout

are that the user can indicate in a parameter associated with the data transfer request primitive that the data

should be sent and delivered directly and a user can indicate that the data transfer is *urgent* which means that it should be sent outside of the flow control mechanism otherwise used by TCP. The equivalent in ISO TP4 is called *expedited data*.

TCP is not a simple protocol and to make life more interesting, there is more than one version of TCP variously called *Reno* and so on depending on the variation in the flow control mechanism used. There are many books on the topic, of which those by Stevens[Ste94] are probably the most well known. Table 10.1 simply list the TCP user primitives and their function. The reader who is interested in more detail, including the detail of the various fields shown in the segment layout shown in Fig. 10.4, or the protocol operation is referred to the textbooks cited. The TCP connection establishment and flow control mechanism are based upon those described for the general case in Sec. 10.3.1, 10.3.2 above.

In Table 10.1 the *Type* column, “request” means a user-to-TCP request, and an “indication” or “confirm” means a TCP-to-user response. Table 10.1 merely serves as an illustration of the various TPDU's which,

Primitive	Type	Function
ABORT	request	Close the connection due to error
ACTIVE-OPEN	request	Initiate a connection
ACTIVE-OPEN-W/DATA	request	Initiate a connection and include data with the request
ALLOCATE	request	Increase buffer space for incoming data
CLOSE	request	Close the connection normally
CLOSING	indication	Tell the TCP user that the remote TCP entity has issued a CLOSE
DELIVER	indication	Tell TCP user that data have arrived
ERROR	indication	Indicates an error has occurred
FULL-PASSIVE-OPEN	request	Tell the TCP entity the user is able to accept connection requests from a specified remote site
OPEN-FAILURE	confirm	Tell TCP user the previous ACTIVE-OPEN request failed
OPEN-ID	confirm	Provides the name associated with the connection requested by the ACTIVE-OPEN request. Requires acceptance of the request by the remote site
OPEN-SUCCESS	confirm	Tells TCP user the previous ACTIVE-OPEN request succeeded
SEND	request	Request to send data over the connection
STATUS	request	Request connection status
STATUS-RESPONSE	confirm	Response to STATUS request
TERMINATE	confirm	Tell TCP user the connection has ended
UNSPECIFIED-PASSIVE-OPEN	request	Tell the TCP entity that the user is able to accept connection request from any remote site

Table 10.1: TCP user primitives and their function

not surprisingly do not differ all that much from the ISO TPDU's which we have not listed.

10.5 UNIX Sockets

Nowadays there is a keen interest in the Linux operating system amongst desktop computer users. Linux, or more correctly UNIX, as the generic system is known, was one of the very first operating systems. Not surprisingly then, a set of primitives was designed to be used by applications who wish to communicate remotely using the UNIX system and a mechanism known as *sockets*. The following is a list of the UNIX socket commands and their meaning.

socket The application specifies how to interpret network addresses, the protocol used (TCP is not the only protocol UNIX can interface with), and whether the communication is connection oriented or connectionless. If successful, it returns an integer (descriptor) of the socket.

close Closes a socket.

bind Associates a name to a socket. The name can be distributed to remote sites, thus allowing them to initiate connections with the local site via the socket.

connect Requests a connection with a remote socket. The call must specify the local socket identifier and the remote socket address. This call is needed when a connection-oriented service is required. Connectionless services such as a datagram service do not require a connect.

listen Often used by a server that responds to requests from many remote sites. Since it cannot have multiple connections simultaneously, it listens for incoming connection requests and puts them into a queue for subsequent processing.

accept Used when a server is able to accept a connection request.

setsockopt Gives the application some control over the socket. For example, the application can use this command to define timeout parameters or buffer limits.

getsockopt Requests information regarding the specified socket.

write Sends data via a connected socket. The call must specify the location and length of data and the socket identifier. It does not specify the destination because a previous connect command established the connection.

writew Similar to write except it allows the application to send data that resides in non-contiguous locations.

send Similar to write except it has an additional parameter, a set of bit flags that allow the application to control certain transmission functions. For example, the flags could indicate the presence of urgent data or request that routing tables not be used in order to let the user specify a route.

sendto Similar to send except it is used with unconnected sockets. As such, it requires that the destination address and its length be specified as a parameter.

sendmsg Similar to send except it allows the application to send data that resides in non-contiguous location. It also consolidates several of sendto's parameters into a single structure to make it easier to use.

read, readv, recv, recvfrom, recvmsg The read and receive (recv) commands are similar to their write and send counterparts. They accept data from remote socket and store it.

select Sometimes a server can interface with many sockets. In this case it needs a mechanism to determine which socket are ready for I/O. The select call does this.

getpeername Gets the address of the remote station to which the specified socket connects.

getsockname Gets the local address for the corresponding socket.

The socket is a UNIX construct and the basis for UNIX network input and output. A UNIX application creates a socket when it needs a connection to a network. It then establishes a connection to a remote station via the socket and communicates with the station by reading and writing data to the socket. Details can be found in the book by Kochran and Wood [Koc89], amongst many others.

10.6 Asynchronous Transmission Mode (ATM)

The ATM protocol stack is shown in Fig. 10.5. It consists of the physical layer, the ATM layer, the ATM adaptation layer and higher layers that permit various applications, mainly voice, video and connection oriented as well as connectionless data applications. It is important to note that there is not direct correspondence with the OSI protocol stack and it is incorrect to refer to the ATM layer as the data link layer, for instance. Equally, it is not really clear whether or not ATM has a transport layer. On the one hand, the

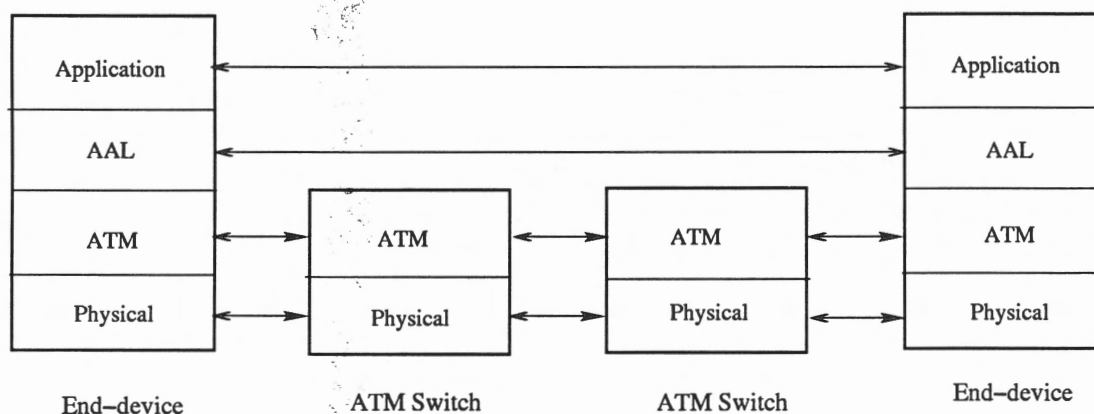


Figure 10.5: The ATM protocol stack

ATM layer has the functionality of a network layer, and there is another layer on top of it (ATM Adaptation Layer or AAL), which sort of makes AAL a transport layer. One of the protocols defined for the Adaptation layer, AAL 5 discussed in Sec. 10.6.5, is functionally similar to UDP, which is unquestionably a transport protocol.

On the other hand, none of the AAL protocols, provide a reliable end-to-end connection as TCP does, although with only very minor changes, they could. Also, in most applications another transport layer is used on top of AAL. Rather than spit hairs, we will discuss the AAL layer and its protocols in this chapter without making a claim that it is a true transport layer.

The AAL layer in ATM networks is radically different than TCP, largely because the designers were primarily interested in transmitting voice and video streams, in which rapid delivery is more important than correct delivery. Remember that the ATM layer just outputs 53-byte cells one after another. It has no error control, no flow control, and no other control. Consequently, it is not well matched to the requirements of most applications.

To overcome this shortfall the ITU I.363 standard defines useful services to application programs, shielding them from the process of chopping data up into cells at the source and re-assembling them at the destination. Different applications have different service requirements, however, which ITU recognised by defining four different classes of service, A, B, C and D as defined in the following table:

Timing Bitrate Mode	A		B	C				D
	Realtime	None	Realtime	None	Realtime	None	Realtime	None
	Constant		Variable		Constant		Variable	
	Connection oriented				Connectionless			

Table 10.2: Service classes supported by AAL

ITU defined AAL 1 through to AAL 4 to cover the 4 service classes. However, it later discovered that the technical requirements for classes C and D were so similar that AAL 3 and AAL 4 became one. None of the protocols pleased industry (the standards, like all standards, were developed by committees in the ATM Forum and ITU and committees usually comprise of people who are no good at doing anything else) and so yet another standard AAL 5 was invented.

10.6.1 Generics of the ATM Adaptation Layer

AAL is divided into two major parts, one of which is often further subdivided. The upper part of the ATM adaptation layer is called the *Convergence Sub-layer (CS)* (some authors call this the *Common Part Sub-layer (CPS)*) which provides the interface to the application. It consists of a sub-part that is common to all applications (for a given AAL protocol) and an application specific sub-part. The function of each of these parts are protocol dependent but can include message framing and error detection.

The convergence sub-layer is furthermore responsible for accepting bit streams or arbitrary length messages from the applications and breaking them into units of 44 to 48 bytes for transmission, adding its own header and trailer as it does so. The exact size is protocol dependent, since some protocol use part of the 48-byte ATM payload for their own headers. At the destination, this sub-layer re-assembles the cells into the original messages. In general, message boundaries are preserved, when present. In other words, if the source sends four 512-byte messages, they will arrive as four 512-byte messages, not one 2048-byte message. For data streams, no message boundaries exist, so they are not preserved.

Below the common convergence sub-layer is the *Segmentation And Reassembly (SAR)* sub-layer. It can add headers and trailers to the data units given to it by the convergence sub-layer to form cell payloads. These payloads are then given to the ATM layer for transmission. At the destination, the SAR sub-layer reassembles the cells into messages. The SAR sub-layer is basically concerned with cells, whereas the convergence sub-layer is concerned with messages.

The SAR sub-layer also has some additional functions for some (but not all) service classes. In particular, it sometimes handles error detection and multiplexing. The SAR sub-layer is present for all service classes but does more or less work, depending on the specific protocol. The communication between the application and AAL layer uses the standard OSI request and indication primitives that we discussed before.

10.6.2 AAL 1

AAL 1 is the protocol used for transmitting class A traffic, that is, real-time, constant bit rate, connection-oriented traffic, such as uncompressed audio and video. Bits are fed in by the application at a constant rate

and must be delivered at the far end at the same constant rate, with a minimum of delay, *jitter*, and overhead. The input is a stream of bits, with no message boundaries. For this traffic, error detecting protocols such as stop-and-wait are not used because the delays that are introduced by timeouts and re-transmissions are unacceptable. However, missing cells are reported to the application, which must then take its own action (if any) to recover from them.

AAL 1 uses a convergence sub-layer and a SAR sub-layer. The convergence sub-layer detects lost and wrongly inserted cells. A cell is wrongly inserted when it is delivered to the wrong destination as a result of an undetected error in its virtual circuit or virtual path identifiers. It also smoothes out incoming traffic to provide delivery of cells at a constant rate. Finally, the convergence sub-layer breaks up the input messages or stream into 46- or 47-byte units that are given to the SAR sub-layer for transmission. At the other end it extracts these and reconstructs the original input. The AAL 1 convergence sub-layer does not have any protocol headers of its own.

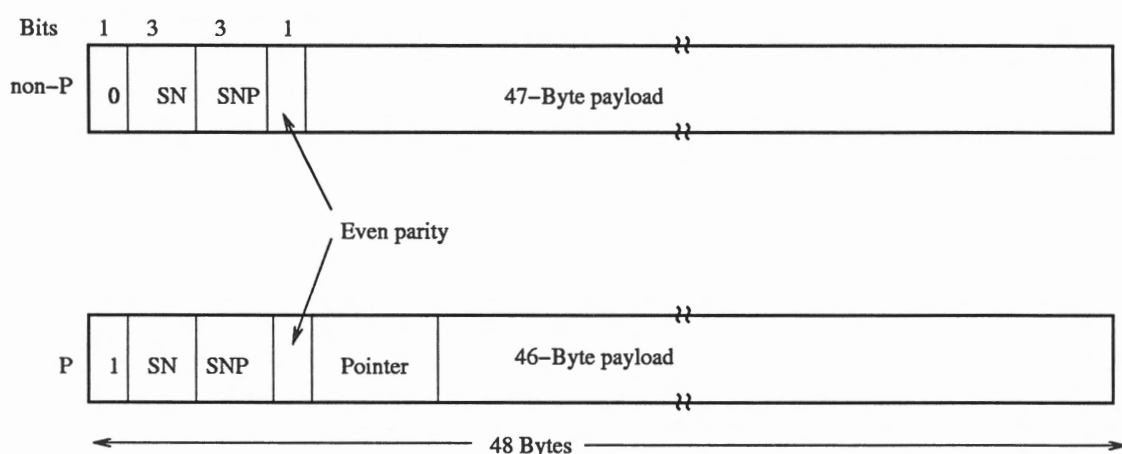


Figure 10.6: The AAL 1 cell format

fits nicely into ATM cell - after 5 bytes are for overhead

In contrast, the AAL 1 SAR sub-layer does have a protocol. The formats of its cells are given in Fig 10.6. Both formats begin with a 1-byte header containing a 3-bit cell sequence number, SN, (to detect missing or wrongly inserted cells. This field is followed by a 3-bit sequence number protection, SNP, (i.e. checksum) over the sequence number to allow correction of single errors and detection of double errors in the sequence field. It uses a cyclic redundancy check (see Sec. 6.4.8 on page 85) with the polynomial $x^3 + x + 1$. An even parity bit covering the header byte further reduces the likelihood of a bad sequence number sneaking in unnoticed. AAL 1 cells need not be filled with a full 47 bytes. For example, to transmit digitized voice arriving at a rate of 1 byte every 125 sec, filling a cell with 47 bytes means collecting samples for 5.875 msec. If this delay before transmission is unacceptable, partial cells can be sent. In this case, the number of actual data bytes per cell is the same for all cells and agreed on in advance.

The P cells (see Fig. 10.6) are used when message boundaries must be preserved. The *Pointer* field is used to give the offset of the start of the next message. Only cells with an even sequence number may be P cells, so the pointer is in the range 0 to 92, to put it within the payload of either its own cell or the one following it. Note that this scheme allows messages to be an arbitrary number of bytes long, so messages can be run continuously and need not align on cell boundaries.

The high-order bit of the *Pointer* field is reserved for future use. The initial header bit of all the odd-numbered cells forms a data stream used for clock synchronization.

10.6.3 AAL 2

AAL 1 is designed for simple, connection-oriented, real-time data streams without error detection, except for missing and wrongly inserted cells. For pure uncompressed audio or video, or any other data stream in which having a few garbled bits once in a while is not a problem, AAL 1 is adequate.

For compressed audio or video, the rate can vary strongly in time. For example, many compression schemes transmit a full video frame periodically and then send only the differences between subsequent frames and the last full frame for several frames. When the camera is stationary and nothing is moving, the differences in frames are small, but when the camera is panning rapidly, they are large. Also message boundaries must be preserved so that the start of the next full frame can be recognized, even in the presence of lost cells or bad data. For these reasons, a more sophisticated protocol is needed. AAL 2 has been designed for this purpose.

As in AAL 1, the CS sub-layer does not have a protocol but the SAR sub-layer does. The SAR cell format is shown in Fig. 10.7. It has a 1-byte header and a 2-byte trailer, leaving room for up to 45 data bytes per

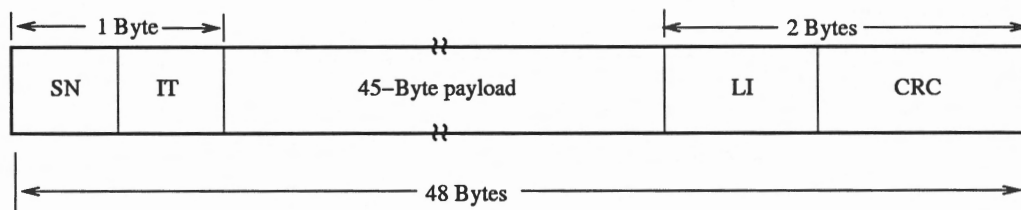


Figure 10.7: AAL 2 cell format

cell. The SN field (Sequence Number) is used for numbering cells in order to detect missing or wrongly inserted cells. The IT field (Information Type) is used to indicate that the cell is the start, middle, or end of a message. The LI (Length indicator) field tells how big the payload is, in bytes (it might be less than 45 bytes). Finally, the CRC field is a checksum over the entire cell, so errors can be detected.

Strange as it may sound, the field sizes are not included in the standard. According to one insider, at the very end of the standardization process the committee realized that AAL 2 had so many problems that it should not be used. Unfortunately, it was too late to stop the standardization process. They had a deadline to meet. In a last ditch effort, the committee removed all the field sizes so that the formal standard could be issued on time, but in such a way that nobody could actually use it. Such is life in the world of standardization.

10.6.4 AAL 3/4

Originally, ITU had different protocols for classes C and D, connection-oriented service and connectionless service for data transport that is sensitive to loss or errors but is not time dependent. Then ITU discovered that there was no real need for two protocols, so they were combined into a single protocol, AAL 3/4.

AAL 3/4 can operate in two modes: stream or message. In message mode, each call from the application to AAL 3/4 injects one message into the network. The message is delivered as such, that is, message boundaries are preserved. In stream mode the boundaries are not preserved. The discussion below will concentrate on message mode. Both reliable and unreliable transport are available in each mode.

A feature of AAL 3/4 not present in any of the other protocols is multiplexing. This aspect of AAL 3/4 allows multiple sessions (e.g., remote logins) from a single host to travel along the same virtual circuit and be separated at the destination. The reason that this facility is desirable is that carriers often charge for each

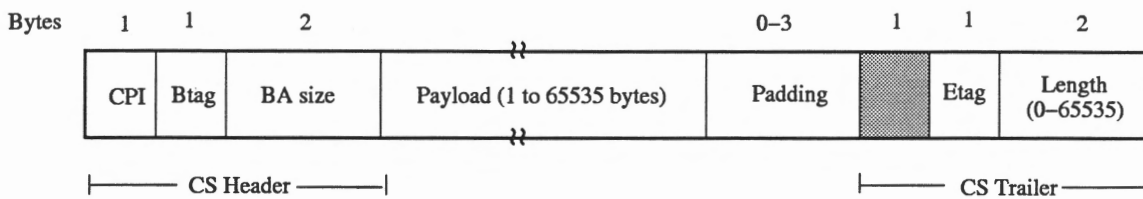


Figure 10.8: AAL 3/4 convergence sub-layer message format

connection setup and for each second that a connection is open. If a pair of hosts have several sessions open simultaneously, giving each one its own virtual circuit will be more expensive than multiplexing all of them onto the same virtual circuit. If one virtual circuit has sufficient bandwidth to handle the job, there is no need for more than one. All sessions using a single virtual circuit get the same quality of service, since this is negotiated per virtual circuit.

This issue is the real reason that there were originally separate AAL 3 and AAL 4 formats: the Americans wanted multiplexing and the Europeans did not. So each group went off and made its own standard. Eventually, the Europeans decided that saving 10 bits in the header was not worth the price of having the United States and Europe not be able to communicate. For the same money, they could have stuck to their guns and we would have had four incompatible AAL standards (of which one is broken) instead of three.

Unlike AAL 1 and AAL 2, AAL 3/4 has both a convergence sub-layer protocol and a SAR sub-layer protocol. Messages as large as 65,535 bytes come into the convergence sub-layer from the application. These are padded out to a multiple of 4 bytes. Then a header and a trailer are attached, as shown in Fig. 10.8.

The CPI field (Common Part Indicator) gives the message type and the counting unit for the BA size and Length fields. The Btag and Etag fields are used to frame messages. The two bytes must be the same and are incremented by one on every new message sent. This mechanism checks for lost or misinformed cells. The BA size field is used for buffer allocation. It tells the receiver how much buffer space to allocate for the message in advance of its arrival. The Length field gives the payload length again. In message mode, it must be equal to BA size, but in stream mode it may be different. The trailer also contains 1 unused byte.

After the convergence sub-layer has constructed and added a header and trailer to the message, as shown in Fig. 10.8, it passes the message to the SAR sub-layer, which chops the message up into 44-byte chunks. Note that to support multiplexing, the convergence sub-layer may have several messages constructed internally at once and may pass 44-byte chunks to the SAR sub-layer first from one message, then from another, in any order. The SAR sub-layer inserts each 44-byte chunk into the payload of a cell whose format is shown

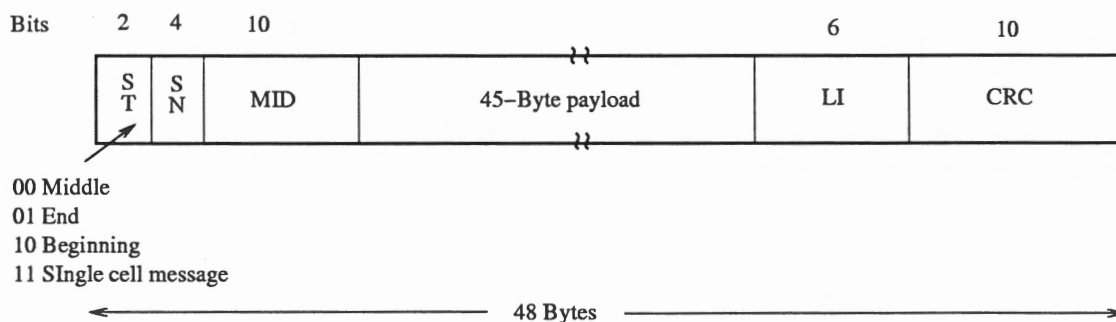


Figure 10.9: AAL 3/4 cell format

in Fig. 10.9. These cells are then transmitted to the destination for re-assembly, after which checksum verification is performed and action taken if need be.

The fields in the AAL 3/4 cell are as follows. The ST (Segment Type) field is used for message framing. It indicates whether the cell begins a message, is in the middle of a message, is the last cell of a message, or is a small (i.e. single cell) message. Next comes a 4-bit sequence number, SN, for detecting missing and wrongly inserted cells. The MID (Multiplexing ID) field is used to keep track of which cell belongs to which session. Remember that the convergence sub-layer may have several messages, belonging to different sessions, buffered at once, and it may send pieces of these messages in whatever order it wishes. All the pieces from messages belonging to session x carry x in the MID field, so they can be correctly re-assembled at the destination. The trailer contains the payload length and cell checksum.

Notice that AAL 3/4 has two layers of protocol overhead: 8 bytes are added to every message and 4 bytes are added to every cell which is somewhat expensive in overhead incurred, especially for short messages.

10.6.5 AAL 5

The AAL 1 through AAL 3/4 protocols were largely designed by the telecommunications industry and standardized by ITU without a lot of input from the computer industry. When the computer industry finally woke up and began to understand the implications of Fig. 10.9, a sense of panic set in. The complexity and inefficiency generated by two layers of protocol, coupled with the surprisingly short checksum (only 10 bits), caused some researchers to invent a new adaptation protocol. It was called *SEAL (Simple Efficient Adaptation Layer)*, which after some discussion, the ATM Forum accepted and assigned it the name AAL 5.

AAL 5 offers several kinds of service to its applications. One choice is reliable service with guaranteed delivery and flow control to prevent overruns. Another choice is unreliable service with no guaranteed delivery, with options to have cells with checksum errors either discarded or passed to the application anyway (but reported as bad). Both uni-cast and multi-cast are supported, but multi-cast does not provide guaranteed delivery.

Like AAL3/4, AAL 5 supports both message mode and stream mode. In message mode, an application can pass a datagram of length 1 to 65,535 bytes to the AAL layer and have it delivered to the destination, either on a guaranteed or a best efforts basis. Upon arrival in the convergence sub-layer, a message is padded out and a trailer added, as shown in Fig. 10.10. The amount of padding (0 to 47 bytes) is chosen to make

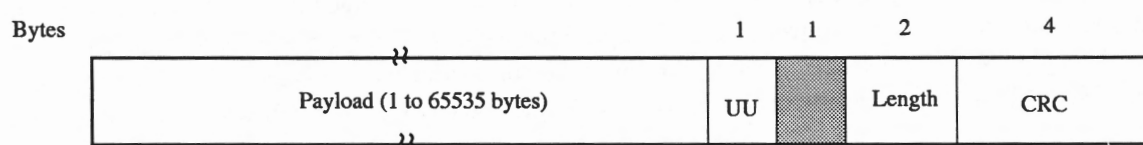


Figure 10.10: AAL 5 convergence sub-layer message format

the entire message, including the padding and trailer, be a multiple of 48 bytes. AAL 5 does not have a convergence sub-layer header, just an 8-byte trailer.

The UU (User to User) field is not used by the AAL layer itself. Instead, it is available for a higher layer for its own purposes, for example, sequencing or multiplexing. The higher layer in question may be the service-specific subpart of the convergence sub-layer. The Length field tells how long the true payload is, in bytes, not counting the padding. A value of 0 is used to abort the current message in mid-stream. The CRC field is the standard 32-bit checksum over the entire message, including the padding and the trailer (with the CRC field set to 0). One 8-bit field in the trailer is reserved for future use.

The message is transmitted by passing it to the SAR sub-layer, which does not add any headers or trailers. Instead, it breaks the message into 48-byte units and passes each of these to the ATM layer for transmission.

It also tells the ATM layer to set a bit in the PTI (Pay load Type Indicator) field of the last ATM cell in order to preserve message boundaries. This violates the most basic principle of protocol engineering, namely that one layer should not interfere with the functions of another. Life is not perfect.

The principal advantage of AAL 5 over AAL3/4 is the much greater efficiency. While AAL 3/4 adds only 4 bytes per message, it also adds 4 bytes per cell, reducing the payload capacity to 44 bytes, a loss of 8 percent on long messages. AAL 5 has a slightly large trailer per message (8 bytes) but has no overhead in each cell. The lack of sequence numbers in the cells is compensated for by the longer checksum, which can detect lost, wrongly inserted, or missing cells without using sequence numbers.

For more information about AAL 5 and how it differs from AAL 3/4, see Suzuki [Suz94].

10.7 Exercises

Exercise 10.1 *Why does the maximum packet life have to be large enough to ensure that not only the packet, but also its acknowledgements have vanished?*

Exercise 10.2 *A protocol uses a 2-way handshake rather than the 3-way one we described to set up connections. Are deadlocks possible? Give an example or prove that none exist.*

Exercise 10.3 *Discuss the advantages and disadvantages of credit versus sliding window protocols.*

Exercise 10.4 *One way of avoiding that packets loitering around the net show up late with a duplicate sequence number, is to use a huge 64-bit sequence number. An optical channel can have a capacity of 64 Tbps. What maximum lifetime is required to make sure that a 12 Tbps network does not have wrap-around problems even with a 64-bit sequence number? Assume a packet size of 128 bits.*

Exercise 10.5 *A TCP machine is sending windows of size 65535 bytes over a 1-Gbps channel that has a 10-msec. one-way delay. What is the maximum throughput achievable?*

Exercise 10.6 *In a network that has a maximum TPDU size of 128 bytes, a maximum TPDU lifetime of 30 sec. and an 8-bit sequence number, what is the maximum data rate per connection?*

Exercise 10.7 *A 1500-byte user-PDU is transported over AAL5.*

1. *How many bytes of padding will be added?*
2. *How many cells are required to carry the resulting CPS-PSDU?*
3. *What is the total overhead (i.e., additional bytes) associated with the user-PDU?*

Bibliography

- [Tom75] Tomlinson, R.S., "Selecting sequence numbers", *Proc. SIGCOMM/SIGOPS Interprocess Communication Workshop*, ACM, pp. 11 – 23, 1975.
- [Tsa84] Stallings, W.A., "Local Network Performance", *IEEE Communications Magazine*, **22**, 2, February 1984.
- [Ste94] Stevens, W.R. "TCP Illustrated, Volumes 1 and II", *Professional Computing Series*, Academic Press, 1994.
- [Koc89] Kochran, S.G. and Wood, P.H. "UNIX Networking", Hayden Books, Indianapolis, 1989.
- [Bin75] Binder, R., "A Dynamic Packet Switching System for Satellite Broadcast Channels", *Proc. ICC*, pp. 41-1 to 41-5a, 1975.
- [Boo77] Boorstyn, R.R., and Frank, H., Large-Scale Network Topological Optimization, *IEEE Trans. Commun.*, **COM-25**, pp. 29-47, Jan. 1977.
- [Chl76] Chlamtac, I., "Radio Packet Broadcast Computer Network – The Broadcast Recognition Access Method", M.S. Thesis, Dept. of Mathematical Sciences, Tel Aviv University, 1976.
- [Chu78] Chu, K., "A Distributed Protocol for Updating Network Topology Information", Report RC 7235, IBM T.J. Watson Research Center, 1978.
- [Cla78] Clark, D.D., Pogran, K.T., and Reed, D.P., "An Introduction to Local Area Networks", *Proc. IEEE*, **66**, pp. 1497-1517, Nov. 1978.
- [Cro73] Crowther, W., Rettberg, R., Walden, D., Ornstein, S., and Heart, F., "A System for Broadcast Communication, Reservation - Aloha", *Proc. Sixth Hawaii Int. Conf. Syst. Sci.*, pp. 371-374, 1973.
- [Dav72] Davies, D.W., "The Control of Congestion in Packet Switching Networks", *IEEE Trans. Commun.*, **COM-20**, pp. 546-550, June 1972.
- [Dij59] Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", *Numer. Math.*, **1**, pp. 269 –271, Oct. 1959.
- [GalHan] Gallo, M.A., "Computer Communications and Networking Technologies", ISBN 0-534-37780-7, Brooks Cole, pp 117, 2001.
- [Git76] Gitman, I., Van Slyke, R.M. and Frank, H., "Routing in Packet-Switching Broadcast Radio Networks", *IEEE Trans. Commun.*, **COM-24**, pp. 926-930, Aug. 1976.

- [Gra78] Grange, J.-L., and Mussard, P., "Performance Measurement of Line Control Protocols in the CIGALE Network", Proc. Computer Network Protocols Symp., Universite de Liege, Liege, Belgium, p. 13, Feb. 1978.
- [Han79] Hansen, L.W., and Schwartz, M., "An Assigned-Slot Listen-Before-Transmission Protocol for a Multiaccess Data Channel", *IEEE Trans. Commun.*, COM-27, pp. 846-857, June 1979.
- [Hon91] Hong, D. and Suda, T., "Congestion Control and Prevention in ATM Networks", *IEEE Network*, pp. 10-16, July 1991.
- [itu321] , ITU: Recommendation I.321, "B-ISDN Protocol Reference Model and its Application, ITU-T, Geneva, 1993.
- [Liu78] Liu, M.T., "Distributed Loop Computer Networks", in *Advances in Computers*, M.C. Yovits (Ed.), New York, Academic Press, pp. 163-221, 1978.
- [Met76] Metcalfe, R.M., and Boggs, D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. ACM*, 19, pp. 395-404, July 1976.
- [Pie72] Pierce, J., "How Far Can Data Loops Go?", *IEEE Trans. Commun.*, COM-20, pp. 527-530, June 1972.
- [Pet62] Petri, A., "Kommunikation mit Automaten", Doctoral Dissertation, 1962.
- [Rud76] Rudin, H., "On Routing and Delta Routing, A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks", *IEEE Trans. Commun.*, COM-24, pp. 43-59, Jan. 1976.
- [Tob74] Tobagi, F.A., "Random Access Techniques for Data Transmission over Packet Switched Radio Networks", Ph.D. Thesis, Computer Science Dept., UCLA, 1974.
- [Tok77] Tokoru, M., and Tamaru, K., "Acknowledging Ethernet", *Compcon*, pp. 320-325, Fall 1977.
- [Suz94] Suzuki, T., "ATM Adaptation Layer Protocol", *IEEE Communication Magazine*, Vol. 32, April 1994, pp. 80-83.
- [Bar64] Baran, P., "On Distributed Communication Networks", *IEEE Trans. Commun. Syst.*, CS-12, pp. 1-9, March 1964.
- [Ros89] Floyd E. Ross, "An Overview of FDDI: The Fiber Distributed Data Interface", *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, September 1989
- [Bur86] W. E. Burr, "The FDDI Optical Data Link", *IEEE Communications Magazine*, Vol. 24, No. 5, May 1986
- [Ros86] Floyd E. Ross, "FDDI - a Tutorial", *IEEE Communications Magazine*, Vol. 24, No. 5, May 1986
- [Petri62] Carl Adam Petri, "Kommunikation mit Automaten", PhD Thesis, Universität Bonn, 1962
- [BauK95] Falko Bause and Pieter Kritzinger, "Introduction to the Theory of Stochastic Petri nets", Vieweg Verlag, Germany, 1995.
- [Dyk88] Doug Dykeman and Werner Bux, "Analysis and Tuning of the FDDI Media Access Control Protocol", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 6, July 1988

- [Eck91] Eckberg, A.E., Doshi, B.T. and Zoccolill, R., "Controlling Congestion in B-ISDN/ATM: Issues and Strategies", *IEEE Network*, Vol. 29, No. 9, 1991
- [Sev87] Kenneth C. Sevcik and Marjory J. Johnson, "Cycle Time Properties Of The FDDI Token Ring Protocol", *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 3, March 1987
- [Joh87] Marjory J. Johnson, "Proof that Timing Requirements of the FDDI Token Ring Protocol are Satisfied", *IEEE Transactions on Communications*, Vol. COM-35, No. 6, June 1987
- [Bau92] Falko Bause, Pieter Kritzinger, Michael Sczittnick, "A Queue Place Net Model of the FDDI Algorithm", *Technical Report*, August 1992
- [Han91] Rainer Händel and Manfred N. Huber, "Integrated Broadband Networks: An Introduction to ATM-based Networks", *Addison-Wesley Publishing Company*
- [bel89] Ferenc Belina, Dieter Hogrefe, "The CCITT-Specification and Description Language SDL", *Computer Networks and ISDN Systems 16 (1988/89) 311-341*, Elsevier Science Publishers B.V. (North-Holland), 1989.
- [ols94a] Ove Faergemand, Anders Olsen, "Introduction to SDL-92", *Computer Networks and ISDN Systems 26 (1994) 1143-1167*, Elsevier Science Publishers B.V.(North-Holland), 1989.
- [SDLtut] International Engineering Consortium online education, "SDL -specification and description language", <http://www.iec.org/online/tutorials/sdl/>.
- [oclwwww] Object Constraint Language, <http://www-3.ibm.com/software/ad/library/standards/ocl.html>.
- [warwww] Jos Warmer, "The future of UML" <http://www.klasse.nl/ocl/index.html>.
- [hol98] E. Holz, "Application of UML in the SDL Design Process", SAM98 Workshop, Berlin, June 1998 <http://citeseer.nj.nec.com/268645.html>.
- [bjowwww] Morgan Bjorkander, "Graphical Programming using UML and SDL". <http://www.telelogic.com/download/paper/graphicalprogramming.pdf>
- [hoowww] Analysis to design - HOORA . <http://www.hoora.org>
- [1] B Selic, J Rumbaugh, "Mapping SDL to UML", Rational Software white paper, 1999. <http://www.rational.com/products/rosert/prodinfo/reading/sdl2uml13.pdf>
- [2] "Telelogic First Tool Vendor To Provide Full Support For Both UML and SDL; Telelogic Tau Significantly Reduces Time To Market Large Scale Projects", Screaming Media, Business Wire, September 1999. <http://industry.java.sun.com/javaneWS/stories/story2/0,1072,19107,00.htm>
- [3] Action Semantics. <http://www.omg.org/gettingstarted>
- [4] SDL Forum Society <http://www.sdl-forum.org/>
- [5] Rational Software www.rational.com

Index

- 100BASE-FX, 126
- 100BASE-T4, 126
- 100BASE-TX, 126

- AAL, 164
- AAL 5, 188
- AAL1, 184
- adaptive routing, 144
- ADCCP, 101
- address-field, 102
- Advanced Data Communication Control Procedure, 101
- analog signal, 9
- angle, 24
- APD, 11
- ARPANET, 9
- ARQ, 101
- asynchronous, 58
- asynchronous modem, 60
- asynchronous protocols, 60
- asynchronous terminal, 70
- Asynchronous Transfer Mode (ATM), 164
- ATM - VCI, 167
- ATM - VPI, 167
- ATM cell, 166
- ATM layer, 164
- attenuation, 28
- Automatic Repeat Request, 101
- avalanche photodiode, 49

- backbone, 116
- bandwidth, 31
- baseband, 28
- binary exponential backoff algorithm, 123
- Binary Synchronous Communication, 102
- Biphase, 37
- bit length, 114
- bit error rate, 79
- bit stuffing, 103
- bit synchronisation, 103
- bit synchronism, 60

- bit-oriented, 101
- block check characters, 84
- Bluetooth, 53
- Bridge, 134
- broadband, 28
- BSC, 102
- buffer preallocation, 146

- CALL ACCEPTED, 157
- call collision, 73
- CALL CONNECTED, 158
- call progress signals, 73
- carrier frequency, 37
- carrier phase, 37
- CBR, 168
- cell switching, 68
- channel adaptor, 60
- channels, 30
- character synchronisation, 103
- checksum field, 102
- choke packet, 147
- chromatic dispersion, 49
- chromophores, 49
- circuit switching, 66
- cladding, 45
- CLEAR CONFIRMATION, 158
- CLEAR INDICATION, 158
- CLEAR REQUEST, 157
- closed user group, 73
- closed user group, 159
- coaxial cable, 28
- codeword, 81
- collect calls, 159
- column parity, 84
- combined amplitude-phase-shift-keying, 39
- committed information rates, CIR, 152
- common bus, 115
- Common Part Sub-layer, 184
- compounding, 79
- concentrator, 62

- congestion, 145
- connection management, 174
- control field, 102
- convergence sub-layer, 184
- core, 45
- core functions, 152
- CRC-12, 88
- CRC-16, 88
- CRC-CCITT, 88
- crosstalk, 63
- CSMA-CD, 121
- CSMA/CA, 119
- CSMA/CD, 118
- current loop interface, 70
- cyclic redundancy code, 85
- cyclical redundancy coding, 85
- D bit, 159
- data link control protocol, 92
- data link layer, 15, 92
- data transfer state, 157
- data-link-control protocol, 92
- Datagram Identification, 160
- Datagram model, 138
- Datagram service, 161
- Datagram Service Signal, 160
- DCE, Data Circuit-terminating Equipment, 70
- deadlock, 149
- deadlock prevention, 145
- decibels, 30
- delay distortion, 30
- demodulator, 39
- device control protocol, 92
- differential encoding, 34
- Differential Manchester encoding, 37
- digital modulation, 37
- digital signal, 9
- DISC, 107
- discard eligibility bits, 153
- downlink, 54
- DPSK, 121
- DSL, 64
- DTE, Data Terminal Equipment, 70
- DWDM, 64
- E1 line, 40
- echo-suppression, 64
- encryption, 17
- end-to-end data-format, 92
- equalizer, 30
- error detection, 174
- error-correcting code, 80
- error-correcting codes, 82
- error-detecting code, 80
- Ethernet, 114, 121
- ETSI, 18
- expedited data, 181
- explicit path routing, 144
- extended address field, 104
- fast connect, 73
- Fast Ethernet, 126
- fast select, 161
- FEP, 8
- fixed routing, 144
- flag, 102
- FLOW CONTROL, 158
- Flow control, 177
- flow control, 139, 148, 174
- Fourier analysis, 25
- Fourier series, 26
- frame, 15
- Frame relay, 150
- Frequency Division Multiplexing, 63
- frequency-shift-keying, 39
- FSK, 39
- full duplex, 58
- Gateway, 134
- generator polynomial, 85
- GEO satellite, 55
- geosynchronous, 55
- Go-back N, 99
- graded index fibre, 46
- graded-index multimode fibre, 48
- half-duplex, 58
- Hamming code, 83
- Hamming distance, 81
- handshaking, 174
- harmonic, 26
- harmonics, 31
- HDLC, 101
- Header Error Control (HEC), 167
- High-level Data Link Control, 101
- ICMP (Internet Control Message Protocol), 162
- ICMP messages, 162

- IEEE, 119
- IEEE 802.3, 121
- impulse noise, 31
- INCOMING CALL, 157
- Information field, 102
- interactively, 7
- interface, 13
- International Standards Organization, 18
- International Telecommunications Union, 17
- Internet, 9, 107
- INTERRUPT, 158
- intersymbol, 30
- IP, 179
 - IP, Destination address, 162
 - IP, DF, 162
 - IP, Fragment offset, 162
 - IP, Header Checksum, 162
 - IP, Identification, 162
 - IP, IHL, 161
 - IP, MF, 162
 - IP, Options field, 162
 - IP, Protocol field, 162
 - IP, Source address, 162
 - IP, Time to live, 162
 - IP, Total length, 162
 - IP, Type of service, 161
 - IP, Version, 161
- ISDN, 151
- ISO, 13
- ISO reference model, 13
- jam signal, 122
- jitter, 48, 185
- LAN, 121
- LAP, 101
- LAPB, 101
- Laser, 11
- LCP, 108
- LED, 11
- light-emitting diodes, 48
- line control procedure, 92
- Link Access Procedure, 101
- link command, 102
- link control procedure, 92
- Link Control Protocol, 108
- link response, 102
- LLC, 119
- local management interface, LMI, 154
- lockup, 149
- logical connection, 92
- Logical Link Control, 119
- longitudinal parity, 84
- MACA, 133
- Manchester encoding, 10, 37, 121
- message synchronisation, 103
- MFSK, 37
- microwaves, 10
- modal dispersion, 48, 49
- modes, 46
- modulation, 10
- modulator, 37
- monitor station, 130
- MPSK, 39
- multiplexing, 60
- multimode fibre, 46
- multiple frequency-shift-keying, 37
- multiple phase-shift-keying, 39
- NACK, 101
- NCP, 108
- negative acknowledgement, 101
- Network Control Protocol, 108
- Network Interconnection, 134
- network layer, 15, 138
- NRZ-L, 34
- NRZI, 34
- NSAP, 174
- Nyquist's law, 40
- Nyquist's theorem, 40
- off-line, 6
- optic fibres, 44
- optic loss, 49
- Optical Time Division Multiplexing, 63
- OSI reference model, 14
- packet switching, 67
- PAM, 37
- parity, 81
- parity bit, 81
- PDM, 37
- permanent Virtual Circuit, 158
- permanent virtual circuits, 152
- Personal Area Network, 53
- phase, 24
- phase-shift-keying, 39

- physical layer, 14, 69
- physical level, 92
- piggybacking, 96
- PIN, 11
- PIN photodiode, 49
- pipelining, 99
- plesiochronous, 78
- Pockels material, 49
- PPM, 37
- PPP, 107, 108
- presentation layer, 17
- protocol, 11
- protocol data units, 14
- protocol specification document, 14
- PSK, 39
- PSTN, 7
- pulse-duration modulation, 37
- pulse-position modulation, 37
- Quality of Service, 165
- radio frequency, 53
- reassembly lockup, 150
- RECEIVE NOT READY, 106
- RECEIVE READY, 106
- receive sequence count, 105
- receiving window, 98
- reference architecture, 14
- REJECT, 106
- Repeater, 134
- repeater, 28, 45
- reservation bits, 129
- RESET, 157, 158
- reverse charging, 159
- ring maintenance, 129
- Router, 134
- routing algorithm, 143
- row parity, 84
- RS232C Serial Interface, 70
- RTS, 133
- satellite communication, 54
- satellite footprint, 56
- scan time, 131
- SDH, 75
- SDLC, 101
- SDLC, Supervisory Format, 106
- SEAL, 188
- Segmentation And Reassembly, 184
- SELECTIVE REJECT, 106
- selective repeat, 99
- self-clocking, 37
- semi-conductor laser diodes, 48
- send sequence count, 105
- sending window, 98
- sequence number, 94
- sequence numbers, 95
- serial mode, 7
- service, 138
- service definition document, 14
- service parameters, 175
- service primitives, 175
- session layer, 16
- session routing, 143
- Shannon's law, 30
- Shielded Twisted Pair(STP), 27
- simplex, 58
- single-mode fibre, 48
- sliding window, 139
- sliding window protocol, 97
- SLIP, 107
- sockets, 182
- SONET, 75
- step-index multimode fibre, 46
- stop-and-wait, 95
- switched virtual circuits, 152
- synchronous, 58
- Synchronous Data Link Control, 101
- synchronous modem, 60
- synchronous protocols, 60
- synchronous terminal, 70
- T1 line, 40
- TCP, 178
- TCP/IP, 9
- TDM, 60
- Teledesic network, 55
- text compression, 17
- thermal noise, 30
- Three-way handshake, 176
- Time Division Multiplexing, 61
- Time Division Multiplexing, 60
- time share, 6
- token bus, 114
- token ring, 114
- token ring performance, 130
- token-holding time, 128

topology, common bus, 115
topology, ring, 116
topology, star, 116
TP ISO, 174
transmission errors, 79
transmission in optic fibres, 45
transport layer, 16, 138
Transport protocol, 174
transverse parity, 84
TSAP, 174

UDP, 179
Unshielded Twisted Pair, 27
uplink, 54

V.41, 88
V24, 70
V3, 73
V35, 70
VCC, 164
Virtual Circuit model, 138
Virtual Circuit setup, 141
VPC, 165

walk time, 131
WAN, Wide Area Network, 9
WAP, 133
Wavelength Division Multiplexing, 63

X20, 72
X20 BIS, 75
X21, 72
X21 BIS, 75
X96, 73

